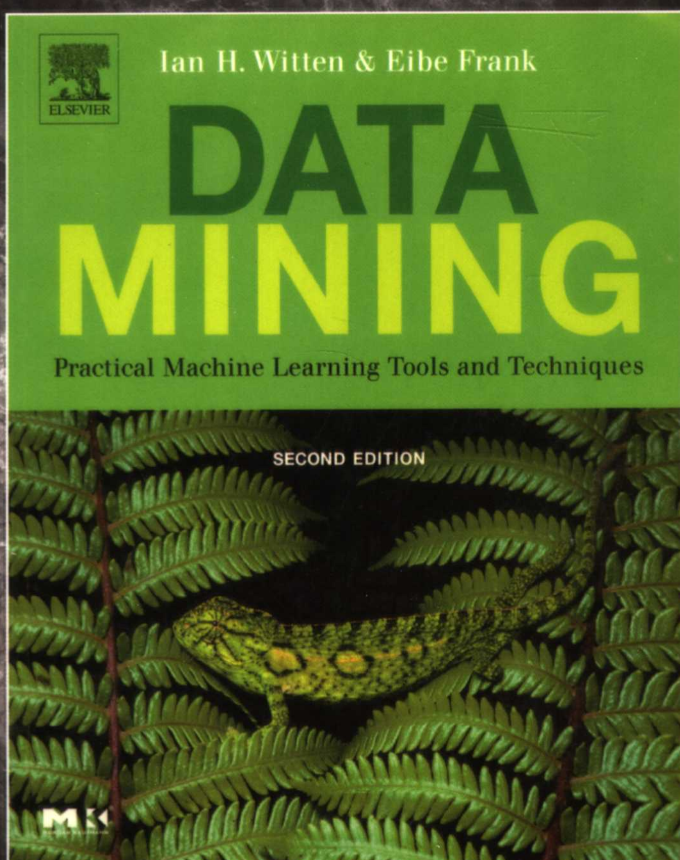
计算机科学丛书

原书第2版

数据挖掘

实用机器学习技术

(新西兰) Ian H. Witten　Eibe Frank 著　董琳　邱泉　于晓峰　吴韶群　孙立骏 译



Data Mining

Practical Machine Learning Tools and Techniques

Second Edition

机械工业出版社
China Machine Press

VISIT…

LANZAROTE Caliente.com

本书以非常容易理解的方式展示了数据挖掘这门新学科：既是用于训练新一代实际工作者和研究者的教科书，同时又能让像我一样的专业人员受益。Witten和Frank热切地追求简单而优美的解决方案。他们将所有的概念都建立在具体实例的基础之上，促使读者首先考虑使用简单的技术。如果简单的技术不足以解决问题，再考虑提升到更为复杂的高级技术。

假如你需要对数据进行分析和应用，本书以及相关的Weka工具包是一个绝佳的起步。

——摘自本书序，Jim Gray（图灵奖获得者）

正如所有受到商业注目的新兴技术一样，数据挖掘的运用也是极其多样化的。言过其实的报导声称可以建立算法，在数据的海洋里发现秘密。但事实上机器学习中没有魔术，没有隐藏的力量，没有炼金术。有的只是一些可以将有用的信息从原始数据中提炼出来的清晰明了的实用技术。本书叙述了这些技术并展示了它们是如何工作的。

本书对1999年的初版做了重大的改动。虽说核心概念没有变化，但本书做了更新，反映出过去五年的变化。新版的重要部分包括了30种新的技术；一个加强了互动界面的Weka机器学习工作平台；有关神经网络的完整信息，一个有关贝叶斯网络的新章节；诸如此类，不胜枚举。

本书提供了机器学习理论概念的完整基础，此外还对实际工作中应用的相关工具和技术提了一些建议，在本书中你将发现：

- 成功数据挖掘技术的核心算法——历经考验的真实技术及领先前沿的方法。
- 转换输入或输出，改善性能的方法。
- 可下载的Weka软件，它集合了能承担数据挖掘任务的机器学习算法，包括对数据进行预处理、分类、回归、聚类、关联规则以及在新的交互式界面上的可视化。

作者简介



Ian H. Witten 新西兰怀卡托大学计算机科学系教授，ACM和新西兰皇家学会成员，曾荣获2004年国际信息处理研究协会(IFIP) 颁发的Namur奖项。他的著作包括《Managing Gigabytes：Compressing and Indexing Documents and Images》、《How to Build a Digital Library》以及众多的期刊和学会文章。



Eibe Frank 新西兰怀卡托大学计算机科学系高级讲师，在机器学习领域有广泛的论文发表，是《Machine Learning Journal》和《Journal of Artificial Intelligence Research》的编委。他还是许多数据挖掘和机器学习学术会议设计委员会成员，也是本书附属的Weka机器学习软件的一位核心开发成员。

ISBN 7-111-18205-7

9 787111 182054

封面设计·陈子平

HZ BOOKS
华章图书

上架指导：计算机/数据挖掘

华章网站 http://www.hzbook.com
网上购书：www.china-pub.com

投稿热线：(010) 88379604
购书热线：(010) 68995259, 68995264
读者信箱：hzjsj@hzbook.com

ISBN 7-111-18205-7/TP・4615
定价：48.00 元

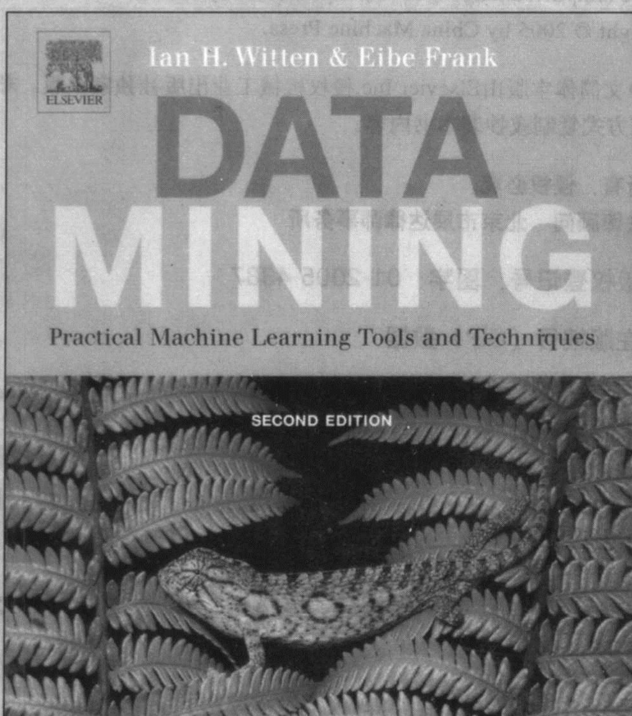
计 算 机 科 学 丛 书

原书第2版

数据挖掘

实用机器学习技术

（新西兰）Ian H. Witten　Eibe Frank 著　　董琳　邱泉　于晓峰　吴韶群　孙立骏 译



Data Mining

Practical Machine Learning Tools and Techniques

Second Edition

机械工业出版社
China Machine Press

本书介绍数据挖掘的基本理论与实践方法。主要内容包括：各种模型（决策树、关联规则、线性模型、聚类、贝叶斯网以及神经网络）以及在实践中的运用，所存在缺陷的分析。安全地清理数据集、建立以及评估模型的预测质量的方法，并且提供了一个公开的数据挖掘工作平台Weka。Weka系统拥有进行数据挖掘任务的图形用户界面，有助于理解模型，是一个实用并且深受欢迎的工具。

本书逻辑严密、内容翔实、极富实践性，适合作为高等学校本科生或研究生的教材，也可供相关技术人员参考。

Ian H. Witten & Eibe Frank: Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (ISBN: 0-12-088407-0).

Authorized translation from the English language edition published by Elsevier Inc.

Copyright © 2005 by Elsevier Inc.

All rights reserved.

Chinese simplified language edition published by China Machine Press.

Copyright © 2005 by China Machine Press.

本书中文简体字版由Elsevier Inc.授权机械工业出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

版权所有，侵权必究。
本书法律顾问　北京市展达律师事务所

本书版权登记号：图字：01-2005-4387

图书在版编目（CIP）数据

数据挖掘：实用机器学习技术（原书第2版）/（新西兰）威滕（Witten, I. H.），（新西兰）弗兰克（Frank, E.）著；董琳等译. －北京：机械工业出版社，2006.2
（计算机科学丛书）
书名原文：Data Mining: Practical Machine Learning Tools and Techniques, Second Edition
ISBN 7-111-18205-7

Ⅰ. 数…　Ⅱ. ① 威…　② 弗…　③ 董…　Ⅲ. ① 数据采集 ② 机器学习　Ⅳ. ① TP274　② TP181

中国版本图书馆CIP数据核字（2005）第157509号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
责任编辑：王　玉
北京京北制版厂印刷 · 新华书店北京发行所发行
2006年2月第1版第1次印刷
787mm × 1092mm 1/16 · 24印张
印数：0 001 -4 000册
定价：48.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线：（010）68326294

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭橥了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum，Stroustrup，Kernighan，Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及庋藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”；同时，引进全美通行的教学辅导书“Schaum's Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业

的教学度身定造的。其中许多教材均已为M. I. T.，Stanford，U.C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

电子邮件：hzjsj@hzbook.com
联系电话：（010）68995264
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

专家指导委员会

（按姓氏笔画顺序）

尤晋元	王　珊	冯博琴	史忠植	史美林
石教英	吕　建	孙玉芳	吴世忠	吴时霖
张立昂	李伟琴	李师贤	李建中	杨冬青
邵维忠	陆丽娜	陆鑫达	陈向群	周伯生
周克定	周傲英	孟小峰	岳丽华	范　明
郑国梁	施伯乐	钟玉琢	唐世渭	袁崇义
高传善	梅　宏	程　旭	程时端	谢希仁
裘宗燕	戴　葵			

译 者 序

无论是数字化管理的需要还是后工业化进程的要求，都使我们日益面对以前无法想像的海量数据。大型消费品公司和商场的市场部门、信用卡公司日复一日地面对如山的销售数据，绞尽脑汁地挖掘其中潜在的市场信号，工业企业的质管部门则需要正确读懂源源不断的质量情况波动报表。毫无疑问，理解乃至最终能够利用这些数据，是值得认真对待的问题。

本书由新西兰怀卡托大学计算机科学系的Ian H. Witten 和Eibe Frank 两位专家集多年的研究和教学成果精心撰写而成。本书（1999年初版）以及配套的Weka 软件一直受到全世界读者和用户的好评。2004年，Witten教授荣获了国际信息处理研究协会（IFIP）颁发的 Namur奖项，这是一个两年一度、用于奖励那些在信息和通信技术的社会应用方面做出杰出贡献及具有国际影响的荣誉奖项。2005年8月，在第11届ACM SIGKDD国际会议上，怀卡托大学的Weka小组荣获了数据挖掘和知识探索领域的最高服务奖，Weka系统得到了广泛的认可，被誉为数据挖掘和机器学习历史上的里程碑，是现今最完备的数据挖掘工具之一（已有11年的发展历史）。在本书第二版的翻译之际，我们欣喜地发现，Weka的每月下载次数已超过万次。本书被很多大学选作专业教材，并在许多学术研究文献中被频繁引用。这些都从侧面验证了本书的杰出成就。

本书的一大特点是能满足不同程度的读者的需求，既有基本理论介绍又有实践应用，读者可以根据需要进行选读而不失连贯性。本书作者Witten教授一再强调，数据挖掘及其不可或缺的技术基础——机器学习，是一个新兴的、充满希望的领域，其应用的前景是极其宽广的。本书既面向有一定专业技术基础、想对技术层面作全面深入了解的读者，也适合于技术基础有限的普通初学者。对于初学者来说，要在短期内尝试入门是难以想像的，然而这部分人群也正是本书要服务的对象。难能可贵的是本书所述的核心技术大都在Weka 系统中得以实现，这不仅可作为学习工具，读者还可以按照自己的需要使用Weka进行数据分析，或在此基础上自行开发创新。

对高等学校的学生来说，本书无疑是一本逻辑严密、内容翔实、极富实践性的教科书；原本对数据挖掘一无所知或有意了解一番的人们，无论是证券专业投资者还是超级市场数据分析员，甚至是面对一大堆彩票数据试图做些什么的业余爱好者，相信我们，读完这本循序渐进、例证充分的入门书籍，再辅之以新西兰怀卡托大学免费下载的、功能强大的Weka系统，成为一个专业水平的数据挖掘者绝非遥不可及的梦想!

我们在翻译的过程中，时时刻刻感受到Witten教授和Frank博士的良苦用心和巨大付出。他们体会到了广大机器学习潜在用户的需求所在，在成功地领导、设计并开发出Weka系统后，于1999年出版了此书的第一版。近年来随着数据挖掘技术的更新和发展，经过Weka研究小组的辛勤工作，Weka软件也日趋成熟完善。两位作者在5年之后重新审视了与Weka软件配套的这本著作，推出了第二版。

本书的翻译是集体完成的，参与翻译工作的有董琳、邱泉、吴韶群、于晓峰、孙立骏，都来自中国，或是对新西兰充满憧憬的新移民，或是负笈留学的莘莘学子，前面四位译者都

曾经或仍然学习、工作于怀大计算机科学系。在翻译过程中，我们无一不被Witten教授和Frank博士的睿智和巨大付出所打动，秉持“信、雅、达”的翻译原则，诚惶诚恐、尽了最大的努力，希望奉献给广大读者一部忠实反映原著风貌的科技书籍。

当然，要翻译好一本书并不是一件容易的事情，我们的水平还很欠缺，本书的翻译一定存在不少问题，还望各位读者批评指教。最后，在翻译过程中，Witten教授和Frank博士对我们的翻译工作给予了许多宝贵的建议和指导，Weka研究小组的成员也对翻译工作给予了很多的关心和支持，我们在此诚致谢意。真诚希望广大中文读者在读完本书后，会认可我们在这白云之乡为大家所尽的绵薄之力。

译 者
2005年9月30日
于新西兰

中文版前言

Nihau（“你好”应为“Nihao”，——译者注）！有机会为新版的《数据挖掘》中文版写上三言两语无疑是很令人高兴的事。我们中的一位 (Ian H. Witten)，从30年前算起曾在武汉、南京、北京等处逗留过，在寥寥几次短暂的中国之行里，亲眼目睹了这个国家从一个只能被描绘成Mamahuhu（“马马虎虎”——译者注）的基础上迅速发展成拥有成熟和先进技术的国家。我们俩人都为本书能对数据挖掘和机器学习领域的关键技术的进一步发展做出贡献而感到欣慰。

西方的作者们时常诧异于偶然发现他们的书籍出现在中国，有时是在中国的再版，有时是完全的中文版。若这次不是我们的学生，我们也将对此一无所知。许多年以前，有人带来一本名为《海量数据管理》的中文版书籍，我们俩中的一位是此书的作者之一。令人遗憾的是，这个翻译本没有被很好地审校，最为突出的一个严重错误是该书的封面上只出现了一位作者的名字——实际上另两位作者在书中的任何部分都没有提到！大家可以想像一下他们的感受。

遇到执著如一的学生是做学问的巨大快乐之一。大约18个月以前，我们的研究生给我们看了《数据挖掘》一书在中国的再版。我们吃惊不已——更让我们吃惊的是随着课程的进展，他们打算将此书翻译成中文。当时，我们的第二版即将完成，我们因此能有机会将中文的翻译工作托付给我们一直以来了解和可信赖的人。

也许可以给作者的最大的肯定莫过于为他们的作品做艰辛的翻译。我们深深地、真诚地感激我们的学生和朋友：董琳、邱泉、于晓峰、吴韶群和孙立骏。感谢他们为此所付出的大量时间和巨大努力。Xiexie（“谢谢”——译者注）！我们知道他们已经理解掌握了书中的内容，因为他们指出了英文版中的一些隐藏的错误，并且向我们提出了一些尖锐的、使我们难以回答的问题。

希望你们能够得益于他们的努力。

英文原文

Nihau! *It is a great pleasure to have the opportunity to write a few words for this new Chinese edition of Data Mining. During a few all-too-short visits to China, beginning thirty years ago with sojourns in Wuhan, Nanjing and Beijing, one of us (IHW) has seen the country blossom in technical maturity and sophistication from a base that all those years ago can only be described as* mamahuhu. *We are both delighted that our book will contribute to further extended growth in the key technologies of data mining and machine learning.*

Western authors are often bemused by occasional sightings of their books in China, sometimes reprints, sometimes complete Chinese editions. We would never know about them were it not for our students. Many years ago someone brought in a Chinese version of a book

co-authored by one of us entitled Managing Gigabytes. *Embarrassingly, this translation had not been properly checked, and a devastatingly prominent error was that only one author's name appeared on the cover of the book-indeed the other two were not mentioned anywhere inside either! You can imagine how they felt.*

Dedicated and committed students are one of the greatest pleasures of being an academic. About eighteen months ago members of our graduate course showed us a reprint of Data Mining *that had been produced in China. We were astonished-even more so because as the course progressed they conceived the idea of actually translating the book into Chinese. At that time we were just completing the second edition, which gave the opportunity of having a Chinese translation by reliable people who we knew well that was bang up to date.*

Perhaps the greatest compliment anyone can pay an author is to painstakingly translate their work for them. We are deeply and sincerely grateful to our students and good friends Lin Dong, Quan Qiu, Xiaofeng Yu, Shaoqun Wu and Lijun Sun for the tremendous amount of time and energy they have put into this project. Xiexie! *We know they have mastered the material because they point out obscure errors in the English edition and ask us penetrating questions that we are hard pressed to answer ourselves.*

We hope you will benefit from their efforts.

Ian H. Witten

Eibe Frank

2005年12月7日

序

技术的发展让我们能够捕获和存储大量的数据。在这些数据集中寻找模式、趋势和异常之处，并且以简单的数量模型归纳之，是当今信息时代的巨大挑战之一——将数据转化为信息，将信息转化为知识。

数据挖掘和机器学习已获得了令人瞠目结舌的进步。统计学、机器学习、信息理论以及计算技术的有机结合，创建了一门具备坚实数学基础和强大工具的完备科学。Witten 和Frank在本书中展示了这个进展的许多方面，辅之以关键算法的实现。因此，这是数据挖掘、数据分析、信息理论以及机器学习多方结合的一个里程碑。如果你在过去的十年里未能追随这个领域（的进步），本书是赶上这个激动人心的进展的绝好机会。如果你一直与该领域同步，那么由Witten 和Frank所撰写的本书及配套的开源工作平台Weka ，可以成为你的工具箱的有用补充。

本书展示了自动从数据中提取模型，然后验证这些模型的基本理论。非常出色地解释了各种模型（决策树、关联规则、线性模型、聚类、贝叶斯网以及神经网），以及如何在实践中运用它们。在这个基础上，讲解了各种方法的实施步骤以及所存在的缺陷。解释了如何安全地清理数据集、如何建立模型，以及如何评估模型的预测质量。本书的大部分是教学指导，但第二部分则全面地描述了系统是如何工作的，并且引导读者游历了作者们在网站上提供的一个公开的数据挖掘工作平台。这个Weka工作平台拥有一个引导读者进行数据挖掘任务的图形用户界面，并拥有出色的、有助于理解模型的可视工具。Weka系统本身是一个有用并深受欢迎的工具，同时又是对本书的一个绝佳补充。

本书以非常容易理解的方式展示了这门新的学科：既是用来训练新一代实际工作者和研究者的教科书，同时又能让像我这样的专业人员受益。Witten和Frank热衷于简单而优美的解决方案。他们对每个主题都采用这样的方法，用具体的实例来讲解所有的概念，促使读者首先考虑简单的技术，当简单的技术不足以解决问题时，便提升到更为复杂的高级技术。

如果你对数据库有兴趣，并且对机器学习方面所知甚少，本书无疑是赶上这个激动人心的技术进步的好机会。如果你有数据要分析和理解，本书和所附的Weka工具将是一个绝佳的起步。

Jim Gray，微软研究院

前　言

计算和通讯的结合建立了一个以信息为源的新领域。但绝大多数信息尚处于它的原始状态：数据。假如数据被定性为记录下的事实，那么信息就是构成数据基础的一系列模式或期望。在数据库中有大量信息被锁定，即那些具有潜在重要性，但尚未被发现和表达出来的信息。我们的任务就是要将它们揭示出来。

数据挖掘是将隐含的、尚不为人所知的，同时又是潜在有用的信息从数据中提取出来，建立计算机程序，自动在数据库中细察，以发现规律或者模式。假如有明显的模式被发现，将可能被归纳以对未来的数据做出准确的预测。当然，问题还是会有的，许多模式可能是陈腐或是没有意义的。另有一些是虚假的，是由于某些具体数据集的偶然巧合而产生的。在现实生活中数据是不完美的：有些部分遭到篡改，有些会丢失。所有发现的东西都是不精确的：任何规律都有例外，任何事例都有规律所不能覆盖到的情况。算法必须稳健得足以应付不完美的数据，提取出不精确但有用的规律。

机器学习为数据挖掘提供了技术基础，用于将信息从数据库的原始数据中提取出来，以可以理解的形式表达、并可以用作多种用途。这是一个抽象的过程：取得数据，据实地推导出数据的结构。本书将介绍在数据挖掘实践中，用以发现和描述数据中的结构模式而采用的机器学习工具和技术。

就像所有新萌发的技术会得到强烈的商业关注一样，数据挖掘的运用也受到大量的技术或大众出版社的追捧。夸大其实的报道声称可以建立学习算法在数据的海洋中遨游来发现秘密。但机器学习中绝没有什么神奇、隐藏的力量，没有炼金术。相反，有的只是一些确实的、简单实用的技术，能将有用的信息自原始数据中提取出来。本书介绍了这些技术并展示了它们是如何工作的。

我们将机器学习解释为从样本中得到结构性的描述。这种描述用于预测、解释和理解。有些数据挖掘应用着重于预测：从数据所描述的过去，预测将来在新的情况下将会发生什么，通常是猜测新的样本分类。但我们同样感兴趣，也许更感兴趣的是“学习”的结果是一个可以用来对样本进行分类的真实结构描述。这种结构描述不仅支持预测，也支持解释和理解。根据我们的经验，在绝大多数数据挖掘实践应用中，用户最感兴趣的莫过于取得对样本实质的把握。事实上，这是机器学习优于传统的统计模型的一个主要优点。

本书解释了许多种机器学习方法，有些是出于教学目的，仅仅罗列了纲要以解释清楚基本的原理。其他的则是实际中的、现在正运用于实际工作的系统。许多方法是时新的，在近几年中发展起来的。

我们创建了一个以Java语言编写的完整的软件资源，用以说明本书中的思想。软件的全名是怀卡托智能分析环境（Waikato Environment for Knowledge Analysis），简称Weka[⊖]，它的源代码可通过国际互联网*www.cs.waikato.ac.nz/ml/weka*得到。它几乎完全产业化地实现了本

⊖ Weka（发音与Mecca押韵）是一种具有奇特天性、不会飞的鸟，只在新西兰发现。

书中介绍的所有技术。它包括机器学习方法的说明性代码和实现技术。针对一些简单技术，提供了清楚而简洁的实现，这些简单技术的设计目的是为了帮助理解机器学习中的运作机制。Weka还提供了一个工作平台，完整、实用、高水准地实现了许多流行的学习方案，这些方案能够直接运用于实际的数据挖掘或研究领域。最后它提供了一个Java类库形式的框架，这个框架支持嵌入式机器学习的应用，乃至新的学习方案的实现。

本书的目的是介绍用于数据挖掘的机器学习工具和技术。读完本书后，你将了解这些技术，并体会它们的功效和可应用性。如果你希望用自己的数据加以实践，用Weka 软件能轻易地做到。

本书跨越了商业书籍上提供的一些在数据挖掘研究案例里非常实际的方法和当前机器学习教科书中的以理论为主的阐述之间存在的鸿沟（在第1章最后的补充读物里简要介绍了这些书）。这个鸿沟是相当大的。为了让机器学习的技术运用得富有成果，需要理解它们是如何工作的。这不是一种可以盲目应用而后期待好结果出现的技术。不同的问题要应用不同的技术，但是哪些技术更适合某种特定情况却不明显：需要知道有可能的解决方案的范围。我们所论及的技术范围非常广泛，这是因为和其他商业书籍不同，本书无意推销某种特定的商业软件或方案。我们列举了大量的例子，但是在例子中所应用的数据集却小得足以跟踪每一步的进展。真实的数据集太大，不能做到这一点（而且通常包含商业秘密）。我们选择的数据集没有展示大型数据集的实际问题，但能够帮助你理解不同技术的作用，它们是如何工作的，以及它们的应用范围是什么。

本书面向对实际工作中的数据挖掘所包含的原理和方法感兴趣的、并有技术基础的普通读者，同样适用于需要获得这方面新技术的信息专家，以及所有希望从技术层面具体理解机器学习包含什么的读者。本书也是为有着一般兴趣的信息系统的实际工作者们所写，例如：程序员、咨询顾问、开发人员、信息技术管理员、规范编写者、专利审核者、好奇的业余爱好者，以及学生和教授。他们需要一本拥有大量实例且简单易读的有关机器学习主要技术是什么，它们做什么，如何运用它，以及它们是如何工作的书。本书面向实际，重点突出“如何做”，同时包括许多算法、代码和实现。所有在实际工作中进行数据挖掘的读者将直接得益于书中叙述的技术。本书目的是帮助试图穿越夸大其词的宣传找出机器学习真谛的人们，也将帮助需要可行的、非学术的、值得信赖的方案的人们。我们避免了对理论和数学知识方面的特殊要求，在页边用浅灰色条形所标记的内容是可选部分，通常是为了照顾对理论和技术感兴趣的读者，跳过这部分内容不会对整体的连贯性有任何影响。

本书分为几个层次，使其既适合对基本概念感兴趣的读者的需要，也能适合想深入详尽地了解、掌握所有技术细节的读者的需要。我们相信机器学习的消费者们需要更多地了解他们运用的算法如何工作。我们常常可以看到，优秀的数据模型与它的诠释者是分不开的，诠释者需要知道模型是如何产生的，从而感知这个模型的长处和局限性。当然，并非所有的使用者都有必要对算法的精妙之处有很深入的理解。

我们把机器学习方法的详尽阐述处理为几个连续部分。最顶层，也就是前三章，读者将学习到基本理论。第1章通过一些例子来说明机器学习是什么，它能用在什么地方。同时提供了一些真实的实际应用。第2、3章叙述了不同的输入和输出，或者称之为知识表达(knowledge representation)。不同的输出要求不同的算法。在下一层的第4章，介绍了机器学习的基本方法，已简化以方便理解。这里原理的给出牵涉到许多算法，但没有涉及到算法所

包含的复杂的细节和精妙的实现方案。为了从机器学习技术的应用升级到解决具体的数据挖掘问题上，必须对机器学习的效果有一个评估。第5章可以单独进行阅读。它能帮助读者评估从机器学习中得到的结果，解决性能评估中出现的某些复杂的问题。

在最底层也是最详细的一层，第6章完全详尽地揭示了实现整系列机器学习算法的步骤，以及在实际应用中为了更好工作所必需的、较为复杂的部分。尽管一些读者也许想忽略这部分的具体内容，但只有到这一层，我们才涉及到完整的、可运作的、并经过测试的机器学习的Weka实现方案。第7章讨论了一些涉及机器学习输入的实际问题。例如：选择和离散属性，这里我们谈到几个更高级的技术来提炼和组合从不同学习技术得出的结果。第一部分的最后一章是展望未来的发展趋势。

本书叙述了在实际工作中的绝大多数机器学习方法。但是，没有涉及到加强学习（reinforcement learning），因为这在实际的数据挖掘中极少应用，也没有包括遗传算法（genetic algorithm），因为它仅仅是一个优化技术；同样也没有包含关系学习（relational learning）和归纳逻辑程序设计（inductive logic programming），因为它们很少被主流数据挖掘应用采纳。

阐明本书思想的数据挖掘系统在第二部分，这是为了清楚地把理论部分和从实践角度如何使用区分开来。如果你急于分析你的数据，不想被技术细节所困扰，可以直接从第4章跳到第二部分。

选定Java来实现本书的机器学习技术，是因为作为面向对象的编程语言，它允许用统一的界面进行学习方案和方法的前期和后期处理。用Java取代C++，Smalltalk 或者其他面向对象的语言，是因为用Java编写的程序能够运行在大部分计算机上，而不需要重新进行编译，或复杂的安装过程，或者最坏的情形，需要修改源代码。Java程序编译成字节码后，能运行在任何安装了适当解释器的计算机上。这个解释器称为Java虚拟机。Java虚拟机和Java编译器能免费用于所有重要平台。

像所有广泛运用的编程语言一样，Java也受到过批评。虽说本书无意对此说三道四，有些批评无疑是正确的。无论如何，在所有的现有可供选择的、能得到广泛支持的、标准化的及拥有详尽文档的编程语言中，Java似乎是本书的最佳选择。它主要的不足是它的运行速度，或者说它在速度上有缺陷。由于执行前，虚拟机先要把字节码翻译成机器编码，执行一个Java程序要比相应的用C语言编写的程序慢好几倍。如果虚拟机用一个即时编译器，根据我们的经验，结果要慢三到五倍。即时编译器将整个字节码块翻译成机器编码，而不是一个接一个地翻译字节码，所以它的运行速度能够得到大幅度的提高。如果对于你的应用来说，这个速度依然很慢，有些编译器能够跳过字节码这一步，直接将Java程序转换成机器编码。当然这种编码不能运行在其他平台上，以至于牺牲了Java的一个最大的优势。

更新和改写过的部分

我们在1999年完成了本书的第一版。现在，2005年4月，我们正在对本书的第二版进行润饰。数据挖掘和机器学习领域在这些年中已臻成熟。虽说在这一版中基本的核心材料没有发生变化，但我们还是尽可能地利用这个机会更新内容，使它能反映过去五年中的变化。当然，也有错误要更正，我们已将错误收在可以公开查询的勘误表中。事实上，虽说发现的错误少得惊人，我们还是希望在第二版中错误会更少（第二版的勘误表可以从本书的主页

http://www. cs.waikato.ac.nz/ml/weka/book.html 中得到)。我们已从整体上对材料进行了编辑，使这本书跟上时代，征引文献的数量翻番。那些最吸引人的部分也已加入了新的材料，下面是重点介绍。

应广泛要求，我们对神经网络部分进行了全面的补充，在4.6节里增加了感知器（perceptron）以及和它密切相关的 Winnow 算法；6.3节里增加了多层感知器（multilayer perceptron）和反向传播（backpropagation）算法。我们对使用核感知器（kernel perceptron）和径向基函数网络（radial basis function network）得到非线性决策边界的实现方法补充了最新的内容。同样应读者的要求，对贝叶斯网络部分增加了一个新的章节。描述了如何基于这些网络来学习分类器，以及如何用 AD树来有效地实现。

本书第一版中所采用的Weka工作平台已经广泛地使用并受到普遍的欢迎。它现在有了一个交互式界面，令人耳目一新，或者说是由三个独立的交互界面组成，使用方便许多。其中最主要的交互界面是Explorer界面。用户可以通过菜单选择和表单填写形式来访问Weka的所有工具。另外一个是Knowledge Flow界面，用户可以设计对数据流处理的配置。还有一个是Experimenter界面，用户可以对一组数据集应用不同参数设置的机器学习算法来建立自动测试、收集测试结果的统计数据，并对结果做重要性测试。这些界面为成为一个数据挖掘实际工作者降低了门槛。在书中我们给出了如何运用这些界面的详细说明。当然，这本书可以离开Weka软件单独使用。为了强调这一点，我们把所有有关工作台的内容单独放在本书最后的第二部分介绍。

Weka数据挖掘的功效在过去五年里不仅变得日益方便，也日渐强大和成熟。如今，它集合了空前数量的机器学习算法和相关技术。它的发展既是被这个领域的发展所推动，也受到了用户和需求的推动。我们相当了解数据挖掘的实际工作者们在想什么。新版本应该包括哪些内容，我们在这方面的体验是值得和盘托出的。

前几章介绍了一些概要的基础内容，相对来说内容改动较少。在第1章里，我们增加了一些具体领域的应用例子。在第2章里增加了有关新的稀疏数据和一些有关字符串属性和日期属性的内容。第3章增加了对交互式决策树结构的介绍，这是一种有用并有启发性的技术，教你如何对自己的数据手工建立一个决策树。

第4章除了介绍分类的线性决策边界，神经网络的基础知识外，还包括一个新的内容：应用于文档分类的多项贝叶斯模型和 logistic回归。在过去的五年，很多人对基于文本的数据挖掘很感兴趣，我们在第2章介绍字符串属性，在第4章介绍了用于文档分类的多项贝叶斯法，在第7章介绍了文本的转换。第4章介绍了许多关于寻找实例空间的高效数据结构的新知识，*k*D树和新近开发的球树。这两个数据结构用来高效地找出最近邻数据，同时能加速基于距离的聚类。

第5章叙述了机器学习的统计评估原则，没有什么改变。主要的增加，除了用于评估预测功效的Kappa统计量的一个注解外，就是成本敏感学习的更多详细对策。我们描述如何运用分类器，在不考虑成本的情况下做成本敏感的预测；或者在训练过程中将成本考虑进来，建立一个成本敏感模型。我们也论及了流行的成本曲线的新技术。

第6章有几处增加，除了上面提到的有关神经网络和贝叶斯网分类器的内容外，也为成功的RIPPER规则学习器所运用的试探法提供了更多的活生生的细节。我们描述了如何利用模型树为数值预测建立规则。展示了如何将局部加权回归运用到分类问题中。最后，叙述了*X*均值

聚类算法，这是对传统k均值算法的一个巨大改进。

有关输入输出的第7章改动最多，因为这方面是近来机器学习实践发展的重点。我们描绘了新的属性选择方案，例如特征搜索和支持向量机的使用；以及新的组合模型技术如叠加回归、叠加logistic回归、logistic模型树以及选择树等等。完整地阐述了LogitBoost（在首版中提到过但未能解释）。新增加了一节关于有用的数据转换的内容，包括重要部分分析以及文本挖掘和时间序列的转换。我们还涉及了在利用没有标签数据来改进分类，包括联合训练和co-EM方法方面的最新进展。

新版重写了第一部分的最后一章有关新的发展方向和不同的远景，以跟上时代的步伐，现在它包括了诸如对抗性学习及无处不在的数据挖掘等新挑战。

致谢

写致谢部分永远是最令人愉悦的！许多人曾帮助过我们，我们很高兴借这个机会来感谢他们。本书发轫自新西兰怀卡托大学计算机科学系的机器学习小组。我们从这个小组的教职成员处得到了慷慨的鼓励和帮助：John Cleary、Sally Jo Cunningham、Matt Humphrey、Lyn Hunt、Bob McQueen、Lloyd Smith及Tony Smith。特别感谢Mark Hall、Bernhard Pfahringer，尤其是Geoff Holmes、小组的领导者和激励的源泉。所有在机器学习小组工作过的成员都对我们的思想有过贡献：特别要提一下Steve Garner、Stuart Inglis 和 Craig Nevill-Manning，为他们在小组工作初创时的帮助，因为那时成功还显得很渺茫，诸事都那么困难。

用于展示书中思想的Weka系统是本书的一个重要组成部分。它是由两位作者共同构思，并由Eibe Frank及Len Trigg、Mark Hall设计和实现的。机器学习小组的许多成员都在早期做了许多贡献。自从第一版以来，Weka小组已经有了可观的扩大，如此多的人做过贡献以致无法在此向每位一一致谢。我们为Remco Bouckaert 在贝叶斯网络的实现，Dale Fletcher在数据库的诸多方面，Ashraf Kibriya 以及 Richard Kirkby 在举不胜举方面的贡献，Niels Landwehr在logistic模型树方面，Abdelaziz Mahoui在K*的实现，Stefan Mutter在关联规则方面，Gabi Schmidberger和Malcolm Ware在许多不同方面的贡献， Tony Voyle在最小均方回归方面，Yong Wang在Pace回归和M5'的实现，Xin Xu 在*JRip*，logistic回归以及其他许多方面的贡献而表示感谢。我们为所有那些献身于我们小组的人们表示真诚的感激，也同样感谢所有来自怀大机器学习小组以外的帮助。

我们的弱点是身处南半球的一个遥远的角落（但很美），我们珍视每一位我们系的来访者所起的重要作用，他们既是我们的义务宣传员，也启迪了我们的思想。我们要特别提到Rob Holte、Carl Gutwin以及Russell Beale，他们每一位都曾访问过几个月；而David Aha虽然只来过几天，却用他的热情和鼓励在小组的初创期起到了大作用；Kai Ming Ting和我们在第7章的许多主题上一起工作了两年，帮助我们的机器学习走上了正轨。

怀大的学生们在这个项目的发展中起到了极大的作用。Jamie Littin在ripple-down规则和关系学习方面工作。Brent Martin探索基于实例的学习和嵌套的基于实例表达。Murray Fife也为关系学习做了艰辛的工作。Nadeeka Madapathage探究了运用功能语言来表达机器学习算法。其他研究生也给了我们数不胜数的影响，尤其是Gordon Paynter ，YingYing Wen以及Zane Bray，他们都曾和我们一同进行文本挖掘工作。怀大的同事们Steve Jones和Malika Mahoui 也为这个方面以及其他机器学习课题做了大量贡献。近期，我们从来自Freiburg的交流学生处得

益颇多，其中有Peter Reutemann和Nils Weidmann。

Ian Witten还想感谢他以前在卡尔加里大学的学生们，尤其是Brent Krawchuk、Dave Maulsby、Thong Phan以及Tanja Mitrovic，他们都对他在机器学习早期的思想形成有所帮助。同样还有卡尔加里大学的教师Bruce MacDonald、Brian Gaines 以及David Hill，坎特伯雷大学的教师John Andreae。

Eibe Frank感激他以前在卡尔斯鲁厄大学的导师Klaus-Peter Huber（现在SAS学院），是Klaus将机器奇妙的学习能力展现给Eibe。在他的学术旅程中，Eibe还得益于和他的加拿大同行Peter Turney、Joel Martin、Berry de Bruijn，以及德国的Luc de Raedt、Christoph Helma、Kristian Kersting、Stefan Kramer、Ulrich Rückert和Ashwin Srinivasan的学术交流。

Morgan Kaufmann公司的Diane Cerra和Asma Stephan为本书的构架做了辛勤的努力，我们的责任编辑Lisa Royse使整个出版过程极为顺畅。Bronwyn Webster则对怀卡托大学这方面的工作给予了出色的支持。

感谢那些默默无闻的书评家的努力，特别是其中一位对本书提出了好几处中肯及富有建设性的意见，帮助我们做了极大的改进。此外，我们还要向负责加利福尼亚大学，Irvine分校的机器学习数据知识库的图书馆工作人员表示感谢，经他们仔细挑选的数据集在我们的研究里是无价之宝。

我们的研究得到了新西兰研究、科学和技术基金以及新西兰皇家Marsden基金协会的赞助。怀卡托大学计算机科学系也通过各种方法予以了慷慨帮助，我们特别感激Mark Apperley富于启迪的领导和温暖的鼓励。第一版的一部分是两位作者在访问加拿大卡尔加里大学期间完成的，我们感谢那里的计算机科学系的支持，还要感谢参与我们机器学习课程教学实验的学生们在冗长课上的积极和有益的态度。

在写第二版时，Ian得到了加拿大杰出信息学研究学会和南阿尔伯特的Lethbridge大学的帮助，他们提供了所有的作者们梦寐以求的东西——一个有着舒心及快乐环境的安静场所。

最后，也是最重要的，我们要感谢我们各自的家庭和伴侣。Pam、Anna和Nikki都十分明白家里有一个写书的人意味着什么（“不要再干了！”），但还是让Ian继续干下去并写完了这本书。Julie总是那么支持，即便Eibe有时要在机器学习实验室中开夜车，而Immo和Ollig则是闲暇时的最大快乐。不管我们源自何方，加拿大、英格兰、德国、爱尔兰、萨摩亚，新西兰将我们聚在一起并给了我们一个理想的、甚至可以说是诗情画意的场所来完成这项工作。

目录

出版者的话
专家指导委员会
译者序
中文版前言
序
前言

第一部分 机器学习工具与技术

第1章 绪论 ……1
1.1 数据挖掘和机器学习 ……1
1.1.1 描述结构模式 ……2
1.1.2 机器学习 ……4
1.1.3 数据挖掘 ……5
1.2 简单的例子：天气问题和其他 ……5
1.2.1 天气问题 ……5
1.2.2 隐形眼镜：一个理想化的问题 ……7
1.2.3 鸢尾花：一个经典的数值型数据集 ……9
1.2.4 CPU性能：介绍数值预测 ……9
1.2.5 劳资协商：一个更真实的例子 ……10
1.2.6 大豆分类：一个经典的机器学习的成功例子 ……12
1.3 应用领域 ……13
1.3.1 决策包含评判 ……14
1.3.2 图像筛选 ……14
1.3.3 负载预测 ……15
1.3.4 诊断 ……15
1.3.5 市场和销售 ……16
1.3.6 其他应用 ……17
1.4 机器学习和统计学 ……18
1.5 用于搜索的概括 ……19
1.5.1 枚举概念空间 ……19
1.5.2 偏差 ……20
1.6 数据挖掘和道德 ……22
1.7 补充读物 ……23
第2章 输入：概念、实例和属性 ……26
2.1 概念 ……26
2.2 样本 ……28
2.3 属性 ……31
2.4 输入准备 ……33
2.4.1 数据收集 ……33
2.4.2 ARFF格式 ……34
2.4.3 稀疏数据 ……36
2.4.4 属性类型 ……36
2.4.5 残缺值 ……37
2.4.6 不正确的值 ……38
2.4.7 了解数据 ……38
2.5 补充读物 ……39
第3章 输出：知识表达 ……40
3.1 决策表 ……40
3.2 决策树 ……40
3.3 分类规则 ……42
3.4 关联规则 ……45
3.5 包含例外的规则 ……46
3.6 包含关系的规则 ……48
3.7 数值预测树 ……50
3.8 基于实例的表达 ……50
3.9 聚类 ……53
3.10 补充读物 ……54
第4章 算法：基本方法 ……56
4.1 推断基本规则 ……56
4.1.1 残缺值和数值属性 ……57
4.1.2 讨论 ……59
4.2 统计建模 ……59
4.2.1 残缺值和数值属性 ……62
4.2.2 用于文档分类的贝叶斯模型 ……63
4.2.3 讨论 ……65
4.3 分治法：创建决策树 ……65
4.3.1 计算信息量 ……68
4.3.2 高度分支属性 ……69
4.3.3 讨论 ……71

4.4 覆盖算法：建立规则 …………………71
4.4.1 规则与树 …………………72
4.4.2 一个简单的覆盖算法 …………………72
4.4.3 规则与决策列 …………………75
4.5 挖掘关联规则 …………………76
4.5.1 项集 …………………76
4.5.2 关联规则 …………………78
4.5.3 有效地建立规则 …………………80
4.5.4 讨论 …………………81
4.6 线性模型 …………………81
4.6.1 数值预测：线性回归 …………………81
4.6.2 线性分类：Logistic回归 …………………82
4.6.3 使用感知器的线性分类 …………………85
4.6.4 使用Winnow的线性分类 …………………86
4.7 基于实例的学习 …………………88
4.7.1 距离函数 …………………88
4.7.2 有效寻找最近邻 …………………88
4.7.3 讨论 …………………93
4.8 聚类 …………………93
4.8.1 基于距离的迭代聚类 …………………94
4.8.2 快速距离计算 …………………94
4.8.3 讨论 …………………96
4.9 补充读物 …………………96
第5章 可信度：评估机器学习结果 …………98
5.1 训练和测试 …………………98
5.2 预测性能 …………………100
5.3 交叉验证 …………………102
5.4 其他估计法 …………………103
5.4.1 留一法 …………………103
5.4.2 自引导法 …………………103
5.5 数据挖掘方案比较 …………………104
5.6 预测概率 …………………107
5.6.1 二次损失函数 …………………108
5.6.2 信息损失函数 …………………108
5.6.3 讨论 …………………109
5.7 计算成本 …………………110
5.7.1 成本敏感分类 …………………111
5.7.2 成本敏感学习 …………………112
5.7.3 上升图 …………………113
5.7.4 ROC 曲线 …………………114
5.7.5 反馈率-精确率曲线 …………………116
5.7.6 讨论 …………………117
5.7.7 成本曲线 …………………118
5.8 评估数值预测 …………………120
5.9 最短描述长度原理 …………………122
5.10 聚类方法中应用MDL原理 …………………124
5.11 补充读物 …………………125
第6章 实现：真正的机器学习方案 ……126
6.1 决策树 …………………127
6.1.1 数值属性 …………………127
6.1.2 残缺值 …………………128
6.1.3 修剪 …………………128
6.1.4 估计误差率 …………………129
6.1.5 决策树归纳的复杂度 …………………132
6.1.6 从决策树到规则 …………………132
6.1.7 C4.5：选择和选项 …………………133
6.1.8 讨论 …………………133
6.2 分类规则 …………………134
6.2.1 选择测试的标准 …………………134
6.2.2 残缺值，数值属性 …………………135
6.2.3 生成好的规则 …………………135
6.2.4 使用全局优化 …………………137
6.2.5 从局部决策树中获得规则 …………………139
6.2.6 包含例外的规则 …………………141
6.2.7 讨论 …………………143
6.3 扩展线性模型 …………………143
6.3.1 最大边际超平面 …………………144
6.3.2 非线性类边界 …………………145
6.3.3 支持向量回归 …………………146
6.3.4 核感知器 …………………147
6.3.5 多层感知器 …………………149
6.3.6 反向传播法 …………………152
6.3.7 径向基函数网络 …………………156
6.3.8 讨论 …………………157
6.4 基于实例的学习 …………………158
6.4.1 减少样本集数量 …………………158
6.4.2 修剪干扰样本集 …………………158
6.4.3 属性加权 …………………159
6.4.4 推广样本集 …………………160
6.4.5 用于推广样本集的距离函数 ……160

6.4.6 推广的距离函数 ……161
6.4.7 讨论 ……161
6.5 数值预测 ……162
6.5.1 模型树……162
6.5.2 建树 ……163
6.5.3 修剪树……163
6.5.4 名词性属性 ……164
6.5.5 残缺值……164
6.5.6 模型树归纳伪代码 ……165
6.5.7 从模型树到规则 ……167
6.5.8 局部加权线性回归 ……168
6.5.9 讨论 ……169
6.6 聚类 ……169
6.6.1 选择聚类的个数 ……169
6.6.2 递增聚类……170
6.6.3 类别效用……174
6.6.4 基于概率的聚类 ……175
6.6.5 EM 算法 ……177
6.6.6 扩展混合模型 ……178
6.6.7 贝叶斯聚类 ……179
6.6.8 讨论 ……179
6.7 贝叶斯网络 ……180
6.7.1 做出预测……181
6.7.2 学习贝叶斯网络 ……184
6.7.3 算法细节……185
6.7.4 用于快速学习的数据结构 ……186
6.7.5 讨论 ……188
第7章 转换：处理输入和输出 ……189
7.1 属性选择 ……190
7.1.1 独立于方案的选择 ……191
7.1.2 搜索属性空间 ……193
7.1.3 特定方案选择 ……194
7.2 离散数值属性 ……195
7.2.1 无指导离散……196
7.2.2 基于熵的离散 ……197
7.2.3 其他离散方法 ……199
7.2.4 基于熵和基于误差的离散 ……200
7.2.5 离散属性转换成数值属性 ……201
7.3 一些有用的转换 ……201
7.3.1 主分量分析 ……202
7.3.2 随机投影……204
7.3.3 从文本到属性向量 ……205
7.3.4 时间序列……206
7.4 自动数据清理 ……206
7.4.1 改进决策树 ……206
7.4.2 稳健回归……207
7.4.3 侦察异情……208
7.5 组合多种模型 ……208
7.5.1 装袋 ……209
7.5.2 考虑成本的装袋 ……211
7.5.3 随机化……211
7.5.4 提升 ……212
7.5.5 叠加回归……215
7.5.6 叠加logistic回归 ……216
7.5.7 选择树……217
7.5.8 Logistic模型树 ……219
7.5.9 堆栈 ……219
7.5.10 误差纠正输出编码……221
7.6 使用没有类标的数据……222
7.6.1 用于分类的聚类 ……223
7.6.2 联合训练……224
7.6.3 EM和联合训练 ……224
7.7 补充读物 ……225
第8章 继续：扩展和应用 ……228
8.1 从大型的数据集里学习……228
8.2 融合领域知识 ……230
8.3 文本和网络挖掘 ……232
8.4 对抗情形 ……235
8.5 无处不在的数据挖掘……236
8.6 补充读物 ……238

第二部分 Weka机器学习平台

第9章 Weka 简介……241
9.1 Weka中包含了什么 ……241
9.2 如何使用Weka ……242
9.3 Weka的其他应用 ……243
9.4 如何得到Weka ……243
第10章 Explorer界面 ……244
10.1 开始着手……244
10.1.1 准备数据 ……244

10.1.2 将数据载入探索者 ……245
10.1.3 建立决策树 ……246
10.1.4 查看结果 ……248
10.1.5 重做一遍 ……249
10.1.6 运用模型 ……250
10.1.7 运行错误的处理 ……251
10.2 探索“探索者” ……251
10.2.1 载入及过滤文件 ……252
10.2.2 训练和测试学习方案 ……255
10.2.3 自己动手：用户分类器 ……259
10.2.4 使用元学习器 ……259
10.2.5 聚类和关联规则 ……260
10.2.6 属性选择 ……262
10.2.7 可视化 ……262
10.3 过滤算法 ……262
10.3.1 无指导属性过滤器 ……264
10.3.2 无指导实例过滤器 ……267
10.3.3 有指导过滤器 ……268
10.4 学习算法 ……269
10.4.1 贝叶斯分类器 ……271
10.4.2 树 ……271
10.4.3 规则 ……273
10.4.4 函数 ……274
10.4.5 懒惰分类器 ……277
10.4.6 其他的杂项分类器 ……277
10.5 元学习算法 ……278
10.5.1 装袋和随机化 ……278
10.5.2 提升 ……278
10.5.3 合并分类器 ……279
10.5.4 成本敏感学习 ……280
10.5.5 优化性能 ……280
10.5.6 针对不同任务重新调整分类器 ……280
10.6 聚类算法 ……281
10.7 关联规则学习器 ……281
10.8 属性选择 ……282
10.8.1 属性子集评估器 ……282
10.8.2 单一属性评估器 ……283
10.8.3 搜索方法 ……284
第11章 Knowledge Flow界面 ……286
11.1 开始着手 ……286
11.2 知识流组件 ……288
11.3 配置及连接组件 ……289
11.4 递增学习 ……290
第12章 Experimenter界面 ……292
12.1 开始着手 ……292
12.1.1 运行一个实验 ……292
12.1.2 分析所得结果 ……294
12.2 简单设置 ……295
12.3 高级设置 ……295
12.4 分析面板 ……296
12.5 将运行负荷分布到多个机器上 ……298
第13章 命令行界面 ……300
13.1 开始着手 ……300
13.2 Weka的结构 ……300
13.2.1 类，实例和包 ……300
13.2.2 weka.core包 ……301
13.2.3 weka.classifiers包 ……301
13.2.4 其他包 ……304
13.2.5 Javadoc索引 ……305
13.3 命令行选项 ……305
13.3.1 通用选项 ……305
13.3.2 与具体方案相关的选项 ……306
第14章 嵌入式机器学习 ……308
14.1 一个简单的数据挖掘程序 ……308
14.2 讲解代码 ……308
14.2.1 main() ……308
14.2.2 MessageClassifier() ……308
14.2.3 updateData() ……312
14.2.4 classifyMessage() ……313
第15章 编写新学习方案 ……315
15.1 一个分类器范例 ……315
15.1.1 buildClassifier() ……315
15.1.2 makeTree() ……315
15.1.3 computeInfoGain() ……322
15.1.4 classifyInstance() ……322
15.1.5 main() ……322
15.2 与实现分类器有关的惯例 ……323
参考文献 ……325
索引 ……342

第一部分 机器学习工具与技术

第1章 绪 论

人工受精的过程是从妇女的卵巢中收集卵子，在与丈夫或捐赠人的精液结合后产生胚胎，然后从中选择几个移植到妇女的子宫里。关键是要选出那些存活可能性最大的胚胎。选择根据60个左右胚胎特征的记录做出，这些特征包括它们的形态、卵母细胞、滤泡和精液样品。特征属性的数量非常大，胚胎学家很难同时对所有属性进行评估，并结合历史数据得出最终结论：这个胚胎是否能够产生一个活的婴儿。在英格兰的一个研究项目中，研究者探索运用机器学习技术，使用胚胎训练数据的历史记录和结果，来做出这个选择。

每年，新西兰奶牛场主都要面临艰难的商业决策：哪些牛应该留在牧场，哪些需要卖到
屠宰场。随着饲料储备的减少，每年牧场在接近挤奶季节末期时只留下1/5的奶牛。每头牛的 3
生育和牛奶产量的历史数据都会影响这个决定。除此以外还要考虑的因素有：年龄（每头牛都将在八岁后接近生育期的终结）、健康问题、难产的历史数据、不良的性情特征（如尥蹶子、跳栅栏）、在下一个季节里不产牛犊。在过去的几年，几百万头牛中的每一头牛都用700多个属性记录下来。机器学习正是用来考察成功的农场主在做决定的时候需要考虑哪些因素，不是为了使决策自动化，而是向其他人推广这些农场主的技术和经验。

机器学习是从数据中挖掘知识。它是一个正在萌芽的新技术，范围涉及生与死、从欧洲到两极、家庭和事业，正逐渐引起人们的重视。

1.1 数据挖掘和机器学习

我们正在被数据所淹没。存在于这个世界和我们生活中的数据总量似乎在不断地增长，而且没有停止的迹象。个人计算机的普及将那些以前会丢弃的数据保存起来。便宜的、大容量的硬盘推迟了用这些数据做什么的决定，因为我们可以买更多的硬盘来保存数据。无处不在的电子数据记录了我们的决策，如超市里的商品选择；个人的理财习惯；以及收入和消费。我们以自己的方式生活在这个世界上，而每一个行为又成为一条数据库里的记录。如今互联网用信息将我们淹没，我们在网上所做的每一个选择都被记录。所有的这些信息记录了个人的选择，而在商业和企业领域存在着数不清的相似案例。我们都知道我们对数据的掌握了解永远无法赶上数据升级的速度。而且在数据量增加的同时，无情地伴随着人们对它的“理解度”的降低。隐藏在这些数据后的是信息，具有潜在用处的信息，而这些信息却很少被显现出来或者被开发利用。

本书是关于如何在数据中寻找模式，这并不稀奇。人类从一开始，就试图在数据中寻找模式。猎人在动物迁徙的行为中寻找模式；农夫在庄稼的生长中寻找模式；政客在选民的意见上寻找模式；恋人在对方的反映中寻找模式。科学家的工作（像一个婴儿）是理解数据，从数据中找出模式，并用它们来指导在真实世界中如何运作，然后把它们概括成理论，这些

理论能够预测出在新的情况下会发生什么。企业家的工作是要辨别出机会，就是那些可以转变成有利可图的生意的行为中的一些模式，并且利用这些机会。

4 在数据挖掘中，计算机以电子化的形式存储数据，并且能自动地查询数据，或至少扩增数据。这仍算不得新鲜事。经济学家、统计学家、预测家和信息工程师长久以来相信，存在于数据中的模式能够被自动地找到、识别、确认并能用于预测。该理论的最新发展使得由数据中找出模式的机遇剧增。在最近几年，数据库急剧膨胀，如每天记录顾客选择商品行为的数据库，正把数据挖掘带到新的商业应用技术的前沿。据估计，存储在全世界数据库里的数据量正以每二十个月翻一倍的速度增长。尽管很难从量的意义上真正验证这个数字，但是我们可以从质上把握这个增长速度。随着数据量的膨胀，以及利用机器承担数据搜索工作已变得普通，数据挖掘的机会正在增长。世界正越来越丰富多彩，从中产生的数据淹没了我们，数据挖掘技术成为我们洞察构成数据的模式的唯一希望。被充分研究过的数据是宝贵的资源。它能够引导人们去获得新的洞察力，用商业语言讲是获得竞争优势。

数据挖掘就是通过分析存在于数据库里的数据来解决问题。例如，在激烈竞争的市场上，客户忠诚度摇摆问题就是一个经常提到的事例。一个有关客户商品选择以及客户个人资料的数据库是解决这个问题的关键。以前客户的行为模式能够被用来分析并识别那些喜欢选购不同商品，和那些喜欢选择同种商品客户的特性。一旦这些特性被发现，它们将被用于当前实际的客户群中，鉴别出那些善变的客户群体，并加以特殊对待，须知对整个客户群都加以特殊对待的成本是高昂的。同样技术还能够被用来辨别出那些对企业当前提供的服务并不满意，但是有可能对的其他服务感兴趣的客户群，并向他们提供特殊建议，从而推广这些服务。在当代竞争激烈、以客户和服务为中心的经济中，如果数据能够被挖掘，它将成为推动企业发展的原材料。

数据挖掘被定义为找出数据中的模式的过程。这个过程必须是自动的或（通常）半自动的。数据的总量总是相当可观的，但从中发现的模式必须是有意义的，并能产生出一些效益，通常是经济上的效益。

如何表示数据模式？有价值的模式能够让我们在新数据上做出非凡的预测。表示一个模式有两种极端方法：一种是内部结构很难被理解的黑匣子；一种是展示模式结构的透明的匣子，它的结构揭示了模式的结构。我们假设两种方法都能做出好的预测，它们的区别在于被挖掘出的模式能否以结构的形式表现，这个结构是否能够经得起分析，理由是否充分，能否用来形成未来的决策。如果模式能够以显而易见的方法获得决策结构，我们就称它们为结构

5 模式，换句话说，它们能帮助解释有关数据的一些现象。

现在我们可以说，这本书是有关寻找、描述存在于数据里结构模式的技术。我们所涉及的大部分技术已经在被誉为机器学习的领域里开发出来。这里我们首先介绍什么是结构模式。

1.1.1 描述结构模式

“结构模式”是什么？你如何描述他们？用什么形式输入？我们将以说明的形式来回答这个问题，而不是尝试给出正式的、最终的死板的定义。这一章后面将给出很多例子，现在让我们从一个例子入手来体验一下我们正在讲解的内容。

表1-1给出了隐形眼镜的一组数据。这组数据是验光师针对病人的情况做出的诊断：使用软的隐形眼镜，硬的隐形眼镜，或不能佩戴隐形眼镜。我们将在以后详细讨论属性的单独意

义。表中的每一行代表一个例子。下面是有关这个信息的部分结构的描述。

```
If tear production rate = reduced then recommendation = none
Otherwise, if age = young and astigmatic = no
           then recommendation = soft
```

结构描述倒不一定像以上这样以规则的形式来表达。另一种普遍使用的表达方法是决策树，它明确了需要做出的决策顺序以及伴随的建议。

表1-1 隐形眼镜数据

年 龄	视力诊断	散 光	泪流量	推荐镜片
young	myope	no	reduced	none
young	myope	no	normal	soft
young	myope	yes	reduced	none
young	myope	yes	normal	hard
young	hypermetrope	no	reduced	none
young	hypermetrope	no	normal	soft
young	hypermetrope	yes	reduced	none
young	hypermetrope	yes	normal	hard
pre-presbyopic	myope	no	reduced	none
pre-presbyopic	myope	no	normal	soft
pre-presbyopic	myope	yes	reduced	none
pre-presbyopic	myope	yes	normal	hard
pre-presbyopic	hypermetrope	no	reduced	none
pre-presbyopic	hypermetrope	no	normal	soft
pre-presbyopic	hypermetrope	yes	reduced	none
pre-presbyopic	hypermetrope	yes	normal	none
presbyopic	myope	no	reduced	none
presbyopic	myope	no	normal	none
presbyopic	myope	yes	reduced	none
presbyopic	myope	yes	normal	hard
presbyopic	hypermetrope	no	reduced	none
presbyopic	hypermetrope	no	normal	soft
presbyopic	hypermetrope	yes	reduced	none
presbyopic	hypermetrope	yes	normal	none

这是一个非常简单的例子。首先，这个表呈现了所有可能值的组合。属性年龄有3种可能值，属性视力诊断、散光和泪流量分别有2种可能值。所以这个表有24行记录（$3 \times 2 \times 2 \times 2 = 24$）。上面所提到的规则并不是真正从数据中概括出来，而仅是对数据的总结。在多数学习情况下，所给出的样本集是非常不完整的，所以一部分工作是要推广到其他新的样本上。用户可以想像如果从上面的表格中忽略一些属性泪流量的值是reduced的行，仍然可以得出规则: 6

```
If tear production rate = reduced then recommendation = none
```

这个规则可以推广到那些遗失的行，并且能正确地把它们填充到表里去。其次，样本中的每一个属性都指定了一个值。现实的数据集不可避免地存在一些样本，这些样本中的某些属性值因为一些原因而不可知，例如数据没有被测量、丢失或其他原因。第三，上面所提到的规则能正确地对例子进行分类，但是通常情况下，因为数据中存在一些错误或者“干扰”，错误分类的情况会发生在用来训练分类器的数据上。

1.1.2 机器学习

现在我们已经有一些输入和输出的概念，下面我们将转入机器学习的主题。究竟什么是学习？什么是机器学习？这是哲学范畴的问题，在这本书里，我们将不涉及有关哲学的问题，而着重立足于实践。然而，在卷起袖子着手开始研究机器学习之前，值得花一些时间从一些基本的问题入手，弄清其中的微妙之处。我们的字典所给出的“学习”的定义如下：

通过学习、体验或者被教授得到知识；

从信息或观察中得知；

获得记忆；

被告知、查明；

7 *接受指令。*

当涉及计算机的时候，这些定义就存在一些缺陷。对于前两条，事实上不可能检测学习是否完成。我们怎么能知道一台机器是否“拥有知识”？我们也不大可能向机器提出问题；即使我们能，那也只是在测试机器回答问题的能力，而不可能测试它学习的能力。我们又如何知道它是否已经“得知”？有关计算机是否能意识到或有知觉的问题是一个激烈争论的哲学问题。对于后三条定义，用人类的术语来说，我们看到它们做出的贡献局限于“变成记忆”和“接受指导”，这个定义对我们所指的机器学习似乎太简单了，也太被动，对于计算机来说，这些任务太平凡。而我们只对在新情况其性能的改善，或至少其性能所具有的潜力感兴趣。须知你可以通过死记硬背的学习方法来“获得记忆”或“得知”，却没有能力在新的情况下运用新的知识；你也能够得到指导却毫无收益。

以前我们是从可操作的角度上定义机器学习：是从大量的数据中自动或半自动地寻找模式的过程，而且这个模式必须有用的。我们可以用同样的方法为学习建立一个可操作的定义：即，当事物以令其自身在将来表现更好为标准来改变其行为时，它学到了东西。

这个定义将学习和表现而不是知识捆绑在一起。你可以通过观察和比较现在和过去的行为来评估学习。这是一个非常客观看上去也满意得多的定义。

但是仍然存在一些问题。学习是一个有点圆滑的概念。很多事物都能以多种途径改变它们的行为，为使它们能在未来做得更好，但是我们不愿意说它们已经真正学到了。一只舒服的拖鞋就是一个很好的例子。拖鞋学到了脚的形状了吗？当然拖鞋确实改变了它的外形从而使它成为一只很舒服的拖鞋。我们不想称其为学习。在日常语言中，我们往往使用“训练”这个词引申出一个不用大脑的学习。我们训练动物甚至植物，尽管这个概念从训练像拖鞋一类没有生命的事物上得到拓展，但是学习是不同的。学习意味着思考和目的，并且学习必须故意去做一些事。这就是为什么我们不愿说一个葡萄藤学会了沿着葡萄园的架子生长，而说它已经被训练。学习没有目的只能是训练，或进一步说，在学习中，目的是学习者的目的，而在训练中，目的是老师的目的。

因此从计算机的视角出发，以可操作的、性能为指导的原则进一步审视第二种学习的定
8 义时，就存在一些问题。当判断是否真正学到一些东西时，需要去看它是否打算去学，是否其中包含一些目的。当应用到机器上时，它使得概念抽象化，因为我们无法弄清楚人工制品是否能够做出有目的的举动。哲学上有关“学习”真正意味着什么的讨论，就像有关“目的”或“打算”真正意味什么一样充满困难。甚至法院也很难把握企图的含义。

1.1.3 数据挖掘

幸运的是，本书所介绍的学习技术并没有呈现出这种概念上的问题，它们被称为机器学习，并没有真正预示任何有关学习到底是什么的特殊的哲学态度。数据挖掘是一个特殊的主题，它涉及到实践意义上的学习，而不是理论意义上的学习。我们对从数据中找出和表达结构模式的技术感兴趣，这个结构模式能作为一个工具来帮助解释数据，并从中做出预测。数据是以一个样本集的形式出现，例如一些已改变其忠诚对象的客户的样本，或一种特定的隐形眼镜针对某种特殊情况被推荐的样本。输出是以在新的样本上做出预测的形式出现：一个具体客户是否将发生改变的预测，或者在给定的条件下哪种隐形眼镜将被推荐的预测。但是这本书是关于从数据中找出和阐述模式的问题，所以输出将同样包括一个真正的结构描述，这个结构能被用来对未知的实例进行分类从而解释决策。对于性能，它有助于提供一个清楚的所需求知识的表现方法。从本质上说，它反映了以上提到的两种学习的定义：知识的获得和使用知识的能力。

许多学习技术寻找有关学到什么的结构描述，这种描述能变得非常复杂，通常表现为如上所示的一套规则，或是本章将要阐述的决策树。这种描述可以用来解释已经学到什么，解释关于新的预测的基本思想，所以能够被人们所理解。经验表明许多机器学习应用到数据挖掘领域中时，拥有清楚的知识结构是必要的，而结构描述也同样重要，它往往比在新的例子上有良好的表现能力更重要。人们频繁地使用数据挖掘不只为了预测，也为了获得知识。从数据中获得知识听起来像一个好的主意，你也能够做到。下面的内容将指导读者如何去做。

1.2 简单的例子：天气问题和其他

在这本书里，我们将用到很多例子。本书是关于从实例中学习，这种形式就显得格外合适。我们将不断提到一些标准的数据集。不同的数据集往往揭示出新的问题和挑战，在考虑学习方法的时候，对不同的问题的思考对研究有着指导意义，也会增加研究的趣味性。实际上，也十分有必要来研究不同的数据集。这里所用的每一个数据集都由一百多个实例问题组成。我们可以在相同的问题集上对不同的算法进行测试和比较。 9

在这一节所用的例子理想化的简单。数据挖掘的真正应用对象是由几千个，或几十万个，甚至几百万个独立的样本组成。但是当我们解释算法做什么和如何做的时候，我们需要一些简单的例子，它们不但能够抓住问题的本质，而且小的数据量也有助于人们对其中每一个细节问题的理解。我们将在这一节以及这本书里研究这些例子。这些例子比较倾向于“学术性”，它们能够帮助我们理解将要发生什么。一些真正在专业领域里运用的学习技术将在1.3节中讨论，这本书中所提到的其他的例子将在本章的补充读物部分提及。

通常真实的数据集还存在一个问题：私有的属性。没有人愿意与你共享他们的客户和产品选购的数据库，从中让你理解他们的数据挖掘的应用和如何工作的细节。公共的数据是非常宝贵的资源，它们的价值随着就像这本书中提到的数据挖掘技术的发展而急剧增加。这里我们关注的是对数据挖掘方法是如何工作的理解。详细了解数据挖掘工作的细节，有利于我们跟踪数据挖掘方法在真正数据上的操作。这就是为什么我们使用一些简单例子的原因。这些例子虽然是简单的，但是足以表现出真正数据集的特性。

1.2.1 天气问题

天气问题是一个很小的数据集，我们将不断地用该数据集来说明机器学习的方法。这完

全是一个虚构的例子，假设可以进行某种体育运动的天气条件。总的来说，数据集中的样本由一些属性值来表示，这些值是从不同的方面测量样本而得到的。天气问题有4个属性：阴晴(outlook)，温度(temperature)，湿度(humidity)和刮风(windy)，结论是是否能玩(play，not)。

表1-2是天气问题一个最简单的形式。所有的4个属性值都是符号类（symbolic categories），而不是数值。其中，阴晴属性值分别是：sunny、overcast、rainy；温度属性值分别是：hot、mild、cool；湿度属性值分别是：high、normal；刮风属性值是true、false。这些值可以建立36个可能的组合（3×3×2×2=36），其中的14个样本作为输入样本。

表1-2　天气数据

阴　晴	温　度	湿　度	刮　风	玩
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

从这些信息中学到的一组规则，也许不是非常好，但是形式应该如下：

```
If outlook = sunny and humidity = high then play = no
If outlook = rainy and windy = true    then play = no
If outlook = overcast                  then play = yes
If humidity = normal                   then play = yes
If none of the above                   then play = yes
```

这些规则按先后次序判断：首先看第一条规则，如果不适用，就用第二条，如此下去。如果一组规则按次序判断，称为决策列（decision list）。作为决策列来判断时，规则能对表中的所有实例正确地进行分类，如果脱离上下文，单独地使用规则进行判断时，有一些规则将是错误的。例如，规则`if humidity=normal then play=yes`将错分一个样本（在表中检查哪个实例被分错）。毫无疑问，一组规则的意义取决于它是如何被判断的。

表1-3是一个比较复杂形式的天气问题。其中两个属性：温度和湿度的值是数值型的值。这意味着许多学习方案必须对两个属性建立不等式，而不是像上一个例子所描述的简单的相等的测试。这个问题称为数值属性问题（numeric-attribute problem）。因为并不是所有的属性都是数值型，所以也叫做混合属性问题（mixed-attribute problem）。

所以第一个规则应该是以下形式：

```
If outlook = sunny and humidity > 83 then play = no
```

得到一些包含数值测试的规则是一个比较复杂的过程。

到现在为止我们看到的规则是分类规则（classification rules）：他们以是否玩的形式去预

测样本的类。也可以不管它的分类，仅仅寻找一些规则，这些规则和不同的属性值紧密关联，称为关联规则（association rules）。从表1-2（天气数据）中可以找到许多关联规则。下面列举出一些好的关联规则。

```
If temperature = cool                then humidity = normal
If humidity = normal and windy = false then play = yes
If outlook = sunny and play = no      then humidity = high
If windy = false and play = no        then outlook = sunny
                                           and humidity = high.
```

表1-3 带有数值属性的天气数据

阴晴	温度	湿度	刮风	玩
sunny	85	85	false	no
sunny	80	90	true	no
overcast	83	86	false	yes
rainy	70	96	false	yes
rainy	68	80	false	yes
rainy	65	70	true	no
overcast	64	65	true	yes
sunny	72	95	false	no
sunny	69	70	false	yes
rainy	75	80	false	yes
sunny	75	70	true	yes
overcast	72	90	true	yes
overcast	81	75	false	yes
rainy	71	91	true	no

以上列出的规则对于所给的数据的准确率达到100%，它们没有做出任何错误的预测。前两个规则分别适用于数据集中的4个样本，第三个适用于3个样本，第四个适用于2个样本。除此以外还能找出很多其他规则。实际上在天气数据中能找出60多个关联规则能适用于两个以上的样本，并且是完全正确的。如果要寻找小于100%准确率的规则，将会找到更多。原因是关联规则能够“预测”任何属性值，并且能够预测一个以上的属性值，而分类规则仅预测一个特定的类。例如，上面第四个规则预测结论是：阴晴是sunny，湿度是high。 10~12

1.2.2 隐形眼镜：一个理想化的问题

前面介绍的隐形眼镜数据，是通过给出的一些有关病人的信息，来告诉你被推荐的隐形眼镜的类型。注意，这个例子仅为了说明问题，它把问题过于简单化，不能用于实际诊断。

表1-1的第一列给出了患者的年龄，这里需要解释一下：presbyopia是一种远视眼，通常发生在中等年纪的人群；第二列是眼镜的诊断：myope是近视，hypermetrope是远视；第三列显示患者是否散光；第四列是有关眼泪的产生率，这是一个重要的因素，因为隐形眼镜需要泪水润滑。最后一列显示所推荐的隐形眼镜的种类：hard、soft或者none。这个表呈现了所有属性值的组合。

图1-1是从这些信息中学到一组规则的例子。这是一个比较大的规则的集合，但是它们确

实对所有的例子都能进行准确地分类。这些规则是完善的也是确定的：它们为每一个可能的例子给出了一个唯一的诊断。但通常不是这样的。有时在某些情况下没有任何规则可以适用，而有时能够适用的规则不止一个，从而会产生出自相矛盾的建议。所以有时规则有可能会与概率或者权联系在一起，来指出其中的一些规则相对于另一些更重要，或者更可靠。

你也许想知道是否一个小的规则集也有很好的表现。如果的确表现好的话，你是否应该使用小的规则集，为什么？这本书将向我们展示了这一类精确的问题。因为样本构造了一个完整的问题空间（problem space）的集合。规则只是总结了所有给出的信息，并以一个不同的、更精练的方式来表达。尽管规则没有概括，但是它也是一个非常有用的分析问题的方法。人们频繁地使用机器学习技术来洞察数据的结构，而不是对新的数据做出预测。实际上，在机器学习领域，一个永恒的成功的研究过程是以压缩一个海量数据库开始，这个海量数据库的数据容量相当于象棋最后阶段所有可能性的数量，从而最终得到一个大小合理的数据结构。为这个计划所选择的数据结构不是一个规则集，而是一个决策树。

```
If tear production rate = reduced then recommendation = none
If age = young and astigmatic = no and
   tear production rate = normal then recommendation = soft
If age = pre-presbyopic and astigmatic = no and
   tear production rate = normal then recommendation = soft
If age = presbyopic and spectacle prescription = myope and
   astigmatic = no then recommendation = none
If spectacle prescription = hypermetrope and astigmatic = no and
   tear production rate = normal then recommendation = soft
If spectacle prescription = myope and astigmatic = yes and
   tear production rate = normal then recommendation = hard
If age = young and astigmatic = yes and
   tear production rate = normal then recommendation = hard
If age = pre-presbyopic and
   spectacle prescription = hypermetrope and astigmatic = yes
   then recommendation = none
If age = presbyopic and spectacle prescription = hypermetrope
   and astigmatic = yes then recommendation = none
```

13 图1-1　隐形眼镜数据的规则

图1-2以一个决策树的形式展示了关于隐形眼镜数据的结构的表达，在多种用途上是一个更简练、明了的规则表示法，并且有更加便于观察的优势。（然而，与图1-1给出的规则集相比，决策树对两个实例的分类是错误的。）树首先对属性泪流量进行测试，产生的两个分支与两个可能的输出结果相对应。如果属性泪流量是reduced（左支），输出是none；如果是normal（右支），第二个测试是散光属性。最后，无论测试是什么结果，所达到的树的叶子指出了向病人推荐的隐形眼镜的种类。从机器学习的方案来看，什么才是最自然、最容易被理解的输出形式，有关这部

14 分内容将在第3章阐明。

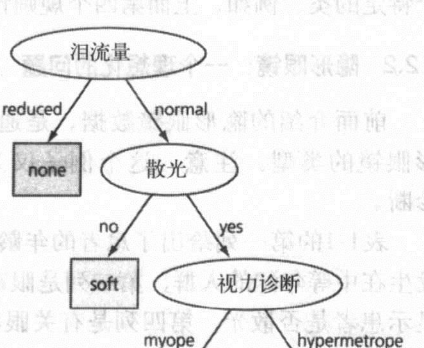


图1-2　隐形眼镜数据的决策树

1.2.3 鸢尾花：一个经典的数值型数据集

鸢尾花数据集是由杰出的统计学家R.A Fisher在20世纪30年代中期开创的，它被公认为用于数据挖掘的最著名的数据集。它包含三种植物种类（鸢尾花 setosa，鸢尾花 versicolor和鸢尾花virginica)、每种各有50个样本。表1-4摘录了这个数据集。它有4个属性组成：花萼长、花萼宽、花瓣长和花瓣宽（测量单位是厘米）。与前面的数据集不同的是，鸢尾花的所有属性都是数值属性。

表1-4 鸢尾花数据

	花萼长(cm)	花萼宽(cm)	花瓣长(cm)	花瓣宽(cm)	种 类
1	5.1	3.5	1.4	0.2	鸢尾花setosa
2	4.9	3.0	1.4	0.2	鸢尾花setosa
3	4.7	3.2	1.3	0.2	鸢尾花setosa
4	4.6	3.1	1.5	0.2	鸢尾花setosa
5	5.0	3.6	1.4	0.2	鸢尾花setosa
…					
51	7.0	3.2	4.7	1.4	鸢尾花versicolor
52	6.4	3.2	4.5	1.5	鸢尾花versicolor
53	6.9	3.1	4.9	1.5	鸢尾花versicolor
54	5.5	2.3	4.0	1.3	鸢尾花versicolor
55	6.5	2.8	4.6	1.5	鸢尾花versicolor
…					
101	6.3	3.3	6.0	2.5	鸢尾花virginica
102	5.8	2.7	5.1	1.9	鸢尾花virginica
103	7.1	3.0	5.9	2.1	鸢尾花virginica
104	6.3	2.9	5.6	1.8	鸢尾花virginica
105	6.5	3.0	5.8	2.2	鸢尾花virginica
…					

下面的规则集是从这个数据集中学到的：

```
If petal length < 2.45 then Iris setosa
If sepal width < 2.10 then Iris versicolor
If sepal width < 2.45 and petal length < 4.55 then Iris versicolor
If sepal width < 2.95 and petal width < 1.35 then Iris versicolor
If petal length ≥ 2.45 and petal length < 4.45 then Iris versicolor
If sepal length ≥ 5.85 and petal length < 4.75 then Iris versicolor
If sepal width < 2.55 and petal length < 4.95 and
   petal width < 1.55 then Iris versicolor
If petal length ≥ 2.45 and petal length < 4.95 and
   petal width < 1.55 then Iris versicolor
If sepal length ≥ 6.55 and petal length < 5.05 then Iris versicolor
If sepal width < 2.75 and petal width < 1.65 and
   sepal length < 6.05 then Iris versicolor
If sepal length ≥ 5.85 and sepal length < 5.95 and
   petal length < 4.85 then Iris versicolor
If petal length ≥ 5.15 then Iris virginica
If petal width ≥ 1.85 then Iris virginica
If petal width ≥ 1.75 and sepal width < 3.05 then Iris virginica
If petal length ≥ 4.95 and petal width < 1.55 then Iris virginica
```

15

这些规则非常繁琐，我们将在第3章里看到能表达同样信息，却更加紧凑的规则。

1.2.4 CPU 性能：介绍数值预测

尽管鸢尾花数据集包含数值的属性，但它的输出——鸢尾花的类型是一个类别，而不是一

个数值。表1-5给出了输出和属性都是数值型的一些数据。它是有关计算机在几个相关属性上处理能力的关联表现。每一行表示一台计算机的配置，总共有209个不同的计算机配置。

表1-5　CPU性能数据

	周期(ns)	主存(KB)		高速缓存(KB)	信道		性能
	MYCT	Min. MMIN	Max. MMAX	CACH	Min. CHMIN	Max. CHMAX	PRP
1	125	256	6000	256	16	128	198
2	29	8000	32000	32	8	32	269
3	29	8000	32000	32	8	32	220
4	29	8000	32000	32	8	32	172
5	29	8000	16000	32	8	16	132
…							
207	125	2000	8000	0	2	14	52
208	480	512	8000	32	0	0	67
209	480	1000	4000	0	0	0	45

处理连续的预测值的传统方法是把结果写成一个线性属性值的和，并为每一个属性加上
16 适当的权。例如：

$$\begin{aligned} PRP = & -55.9+0.0489MYCT+0.0153MMIN+0.0056MMAX \\ & +0.6410CACH-0.2700CHMIN+1.480CHMAX \end{aligned}$$

（缩写的变量名在表的第二行给出。）这个公式称为回归公式（regression equation），决定权值的过程称为回归（regression）。我们将在第4章介绍一个在统计学中众所周知的回归过程。然而，基本的回归方法不足以发现非线性关系（尽管变量确实存在，在6.3节中将阐述其中的一个）。在第3章中，我们将考查能够用于预测数值量的不同表现方式。

在鸢尾花和中央处理器CPU性能数据中，所有属性都是数值属性。而实际的数据通常表现为数值和非数值属性的混合形式。

1.2.5　劳资协商：一个更真实的例子

表1-6是劳工谈判数据集，它概括了加拿大人在1987~1988年劳资协商的结果。它是由那些在商界和个人服务领域之间，为至少有500人的组织（教师、护士、大学老师、警察，等等）达成的集体协议。每一个案例涉及到一个合同，结果是是否合同被认为可以接受（acceptable），或者不能接受（unacceptable）。可以接受的合同是那些劳工和管理双方都能接受的协议。那些不能接受的合同是因为某一个团体不愿接受，而导致提议失败；或者可以接受的合同却在一定范围内引起不安，因为从专家的角度来看，它们是不应该被接受的。

这个数据集有40个样本（另外通常留出17个样本给测试）。与其他表不同，表1-6是以列而不是行的形式来表示样本，否则表的宽度将要延伸好几页。其中一些未知或残缺的值用问号来标记。

这个数据集比我们先前介绍的更真实。它存在很多残缺值，看上去似乎不太可能得到一个真正的分类。

图1-3展示了表示数据集的两个决策树。图1-3a简单且近似：它没有如实完全表现数据。例如，对一些实际上被标注为good的合同，它预测为bad结果。但是它做出了直觉的判断：如

表1-6 劳资协商数据

属 性	类 型	1	2	3	…	40
duration	years	1	2	3		2
wage increase 1st year	percentage	2%	4%	4.3%		4.5
wage increase 2nd year	percentage	?	5%	4.4%		4.0
wage increase 3rd year	percentage	?	?	?		?
cost of living adjustment	{none, tcf, tc}	none	tcf	?		none
working hours per week	hours	28	35	38		40
pension	{none, ret-allw, empl-cntr}	none	?	?		?
standby pay	percentage	?	13%	?		?
shift-work supplement	percentage	?	5%	4%		4
education allowance	{yes, no}	yes	?	?		?
statutory holidays	days	11	15	12		12
vacation	{below-avg, avg, gen}	avg	gen	gen		avg
long-term disability assistance	{yes, no}	no	?	?		yes
dental plan contribution	{none, half, full}	none	?	full		full
bereavement assistance	{yes, no}	no	?	?		yes
health plan contribution	{none, half, full}	none	?	full		half
acceptability of contract	{good,bad}	bad	good	good		good

果第一年工资的增长幅度很小（小于2.5%），这个合同是bad（对雇员来说）。如果第一年工资的增长大于这个百分比，而且还有很多法定假期（超过10天），它就是good。如果法定假期比较少，但是如果第一年工资的增长幅度足够的大（大于4%），它也是good。

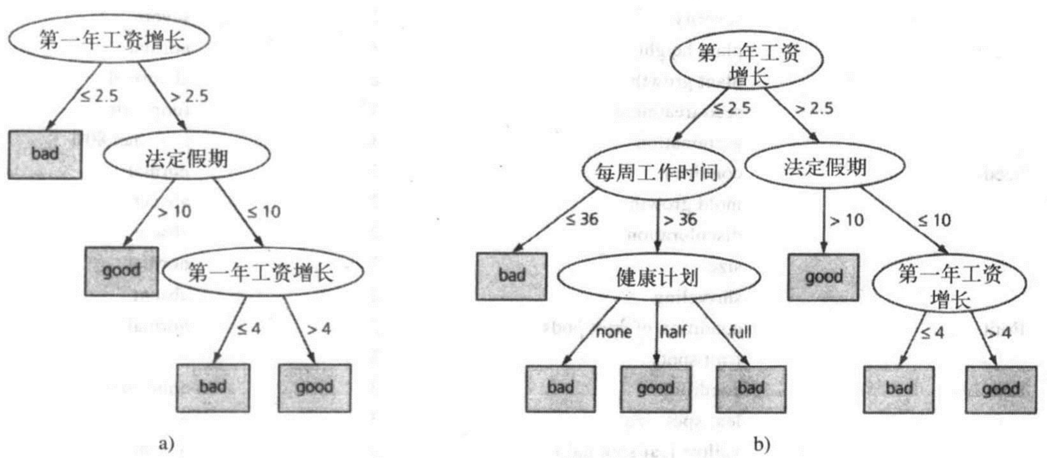


图1-3 劳资协商的决策树

图1-3b是一个更为复杂的表示同样数据集的决策树。实际上，这是一个用来建树的实际数据集的更精确的表达方式。但是它并不一定是好坏合同的概念的一个更准确的表现。顺着左支向下看，直觉上并不能使人理解为什么当工作时间超过36小时时，如果没有给出健康计划或者给出一个完整的健康计划，这个合同就是坏的；但是如果给出了一半的健康计划，这个合同却又变成好的了。健康计划在决策中要起作用是十分合理的，但是不能说制定了一半的健康计划是好的，而拥有完整健康计划和没有健康计划是坏的。这可能是因为建立决策树时使用了人造数据，因而没有真正反映有关好和坏差异的真正特征。

图1-3b所显示的树在分析用于训练分类器的数据时拥有更高的准确性，但是在分析一个独立的测试数据集时，它的性能可能有所降低。原因是在训练数据过程中出现了“过度拟合”(overfitted)。图1-3a是把图1-3b修剪以后得到的树。有关内容我们将在第6章进行学习。

1.2.6　大豆分类：一个经典的机器学习的成功例子

在将机器学习应用于解决实际问题时，经常引用的一个早期的成功案例是为诊断大豆疾病找出鉴别规则。数据是从描述大豆作物疾病的问卷调查表中采集的。总共有680多个样本，每一个样本表示一个有病的大豆作物。大豆苗从35个属性方面被检测，每一个属性拥有一组小范围的可能值。所有样本都由一个植物生物学领域的专家标记上诊断结果。有19个恐怖的疾病分类组成，如腐皮壳菌层茎杆溃疡，丝核菌根腐烂，细菌枯萎病，等等。

表1-7给出了所有属性，每个属性拥有多个不同的值，每一个实例记录一个特定的植株。为了便于阅读，所有属性被划分到不同的分类中。

表1-7　大豆数据

	属　性	值的数量	实例值
Environment	time of occurrence	7	July
	precipitation	3	above normal
	temperature	3	normal
	cropping history	4	same as last year
	hail damage	2	yes
	damaged area	4	scattered
	severity	3	severe
	plant height	2	normal
	plant growth	2	abnormal
	seed treatment	3	fungicide
	germination	3	less than 80%
Seed	condition	2	normal
	mold growth	2	absent
	discoloration	2	absent
	size	2	normal
	shriveling	2	absent
Fruit	condition of fruit pods	3	normal
	fruit spots	5	—
Leaf	condition	2	abnormal
	leaf spot size	3	—
	yellow leaf spot halo	3	absent
	leaf spot margins	3	
	shredding	2	absent
	leaf malformation	2	absent
	leaf mildew growth	3	absent
Stem	condition	2	abnormal
	stem lodging	2	yes
	stem cankers	4	above soil line
	canker lesion color	3	
	fruiting bodies on stems	2	present
	external decay of stem	3	firm and dry
	mycelium on stem	2	absent

（续）

	属 性	值的数量	实例值
	internal discoloration	3	none
	sclerotia	2	absent
Root	condition	3	normal
Diagnosis		19	diaporthe stem canker

下面是两个从数据中学到的规则的例子。

```
If   [leaf condition is normal and
     stem condition is abnormal and
     stem cankers is below soil line and
     canker lesion color is brown]
then
     diagnosis is rhizoctonia root rot

If   [leaf malformation is absent and
     stem condition is abnormal and
     stem cankers is below soil line and
     canker lesion color is brown]
then
     diagnosis is rhizoctonia root rot
```

这些规则很好地展示了预备知识的潜在作用（在机器学习领域通常称为领域知识domain knowledge）。以上两种描述的不同之处在于叶子状态（leaf condition）是正常还是叶子畸形（leaf malformation）缺失（absent）。实际上在这个领域，如果叶子状态是正常，那么叶子就不可能是畸形。所以以上任何一种情况的发生将是另外一种情况的一个特例。因此如果第一个规则是正确的，那么第二个必定是正确的。当叶子不是畸形，但是叶子状态不正常，就要运用第二条规则，也就是说叶子有除了畸形以外的不正常状态存在。这些规则不是从漫不经心的阅读中能显而易见的。

在20世纪70年代末的研究中发现，每一个疾病类型的诊断规则都能由机器学习算法从300个训练样本中建立。这些训练样本是从整个病例中精心挑选出来的，每一个样本都截然不同，并且在样本空间"相隔很远"。与此同时再对那些做出诊断的植物病理学家进行访问，然后把他们的专家经验翻译成诊断的规则。令人惊奇的是，与专家产生的规则相比，计算机产生的 17～21
规则在剩余的测试实例上有更卓越的表现。它对病毒预测的准确率达到97.5%，相对而言，由专家给出的规则的正确率只有72%。此外，不但由学习算法发现的规则胜过合作专家的结果，而且用（学习算法）揭示的规则代替了专家自己的规则，令人惊讶。

1.3 应用领域

我们在开始介绍的例子是猜测性的研究项目，不是产品系统。以上所示的例子都是一些小问题，我们特意选择一些小型的例子，用来研究书中所提到的算法。那么真正地运用在哪里？这里将介绍一些真正进入实际应用的机器学习方法。

在领域应用方面，强调性能方面学习的使用，重点是在新的样本上有良好的表现能力。这本书同时描述了使用学习系统从由数据推断出的决策结构中获得知识。我们相信这与做出高质量的预测一样重要，也许在将来作为技术运用后还会更加重要。但是它在应用领域尚缺乏代表性，因为当使用学习技术来获得洞察力时，结论通常不是一个能运用在实际工作中的应用系统。然而，从下面的三个例子可以看出，决策结构是可以理解的，它是衡量方案是否成功的一个关键特征。

1.3.1 决策包含评判

当你申请贷款，必须填一张有关金融和个人信息的调查表。信贷公司根据这些基本信息做出决策：是否批准你的贷款。这些决策的制定要通过两个阶段完成。首先，运用统计的方法来决定那些明确能“接受”或“拒绝”的案例。剩下是处于临界线的比较复杂的案例，需要人来做出判断。例如，一个信贷公司使用一个统计的决策过程产生一个数值参数，这个参数是从调查表提供的信息中计算出的。如果这个参数超过一个预先设定的标准，申请人将被接受，如果低于标准的下限，申请人将被拒绝。这个方案能够处理90%的案例，剩下的10%需要由信贷员亲自做出决策。对申请人能否真正偿还他们的贷款的历史数据研究表明，有超过一半处于临界的并得到贷款的申请人未按期付款。当然如果拒绝给达到临界条件的申请人
22 贷款，将使问题简单化。但是信贷公司专业人士指出，如果能够可靠地探明那些在未来有偿还能力的客户，那么公司必须争取这些客户。他们通常是信用机构里较活跃的客户，他们的资金长期处于不稳定状态。因此在一个不喜欢坏账的公司会计和一个追求业绩的销售主管之间，必须达成一个合适的妥协。

把这个例子引入机器学习。在这个案例中，输入由1000个训练样本组成，这些样本是处于临界线并得到贷款的案例，而且标明借贷人是否最终偿还了贷款。每一个训练样本由调查表里提取的20多个属性组成，如：年龄、为当前雇主工作的年数、在当前地址居住的年数、拥有银行账户的年数，以及所持有的其他信用卡。用一个机器学习程序生成了一个小的分类规则集，它在一个单独选出的测试集上正确预测了2/3处于临界线的案例。这些规则的使用不但提高了信贷决策的成功率，而且信贷公司发现可以用它们来向客户解释决策背后的原因。尽管这是一个探索中的项目，只取得一些小的发展成果，但是信贷公司显然对得到的结果很高兴，因为所得到的规则能够立刻运用到实际工作中。

1.3.2 图像筛选

从卫星技术发展开始，环境科学家就已经试图从卫星图片上探测出浮油区域，为了能及早给出生态灾难的警报和制止非法倾倒的行为。雷达卫星为监控沿海水域提供了机遇，因为它不受白天、黑夜以及天气条件的影响。浮油区域将以一个深色区域出现在图像上，它的大小、形状的发展变化取决于天气和海洋条件。然而，其他一些看上去比较深色的区域是由于当地的气候条件，如大风造成的。区别出浮油区域是一个代价很高的手工过程，需要接受过培训的人员进入到图片中的每一个区域进行实地调查。

现在，一个危险探测系统已经被开发，它能代替手工完成后续处理，从中筛选图片。目的销售给世界范围内的广大终端用户。因为各个政府机构和公司处于不同地域，有着不同的目标和应用，所以，系统需要高度客户化来适应不同的需求。机器学习用由用户提供的浮油区域和非浮油区域的图像样本来训练系统，并且能让用户控制如何在未能察觉到的浮油区域和虚假警报之间进行权衡。与其他机器学习应用的不同之处在于，这个学习方案本身将投入到实际应用领域里，而其他的则是先产生一个分类器，然后投入到实际运用中。

这里的输入是从雷达卫星获得的一个原始的像素图像集，而输出是一个非常小的图像集，这些图像是推断出的浮油区域，并且用彩色的边框加以标记。在这一过程中，首先需要运用标准图像操作对图像进行标准化处理。接着，识别出可疑的深色区域。从每一个区域提取数
23 十个属性用于刻画区域的规模、形状、面积、色彩饱和度、边界的锐利度和锯齿形状，附近

的其他区域，以及有关的邻近区域的背景信息。最后把标准的学习技术运用在获得的属性矢量上。

在此过程中遇到了一些有趣的问题。第一，用于训练的数据极少。（幸运的是）原油泄漏事件是很少发生的，而人工分类的成本极其昂贵。第二，问题的不均衡性：在训练数据中，极少比例的深色区域是由真正的浮油造成的。第三，样本自然地组成批，每一批都是从一个图像中提取的区域的集合，批与批之间的背景是不一样的。最后作为一个过滤器来完成筛选的工作，并且必须为用户提供一个可变更虚假报警率的简便的方法。

1.3.3 负载预测

在电力供应行业，尽可能早地判断出未来电力的需求量有着重要的意义。如果能准确地估计出每小时、每天、每月、每季和每年的最大和最小负载，那么电力公司能够在很多领域，如运作储备、维护调度，以及燃料库存管理上取得很高的经济性。

在过去十年里，一个自动的负载预测助理已经为一个主要的电力供应商工作了，它能够提前两天提供每小时负载的预测。完成这个预测需要利用过去15年收集的数据手工建立一个复杂的模型。这个模型有三个组件：年基本负载量、年内负载周期以及假期的影响。要得到基本负载就需要将数据做正常化处理，也就是对前一年数据进行标准化，方法是：每小时读出的负载数值减去每小时平均负载量，然后除以全年负载的标准差。电力负载在三个基本频率上显示出周期性的变化：每天，电的用量在早晨最低，而中午和晚上达到最高；每周，电力需求在周末的时候较低；按季节统计，电力需求量在每年的冬季和夏季增加，分别是为了取暖和降温的需要；在一些主要的节假日里，如感恩节、圣诞节和新年，电力负载和平时相比显示出急剧的变化，对于这些特殊时段要分别用过去15年在同一天每小时的平均负载量单独进行建模。一些次要的官方节假日，如哥伦布日，将和学校假期一起作为正常日（消费）模型的分支进行处理。以典型时段的次序综合考虑所有因素重建一年的负载，接着把节假日插入到正确的位置，然后对负载进行反向规格化，从而计算出大致的增长量。

综上所述，所建的负载模型是静态的，是从历史数据中手工建立起来的，并且隐含地假设全年的天气情况“正常”。最后一步，要考虑天气条件的影响，使用一个技术找出历史上和当 24
前情况相似的天，并把那天的历史信息作为一个预测器。这时的预测是对静态负载模型的一次附加修正。为了防止出现较大偏差，需要找出非常相似的8天，取它们附加修正值的平均值。为此需要建立一个数据库记录当地3个气象中心在过去15年里每小时的温度、湿度、风速和云层覆盖度，以及真实负载和由静态模型预测的负载量之间的差异。用一个线性回归分析方法分析相关的参数对负载的影响，并且使用系数对用于寻找最相似日子的距离函数进行权衡。

这个系统表现出和受过培训的专业人员相近的预测能力，但预测的速度更快，只要几秒钟而不是几小时就能对某一天做出预测。操作人员可以通过分析预测结果对天气变化的敏感程度，来验证系统为修正预测结果所使用的天气变化“最相似”的日子。

1.3.4 诊断

诊断是一个专家系统的主要应用领域。虽说手工编写的规则在专家系统中通常运作很好，但有时会过于耗费人工，在这种情况下机器学习就会有用。

对于电机设备，例如，发动机和发电机，预防性的维护能够避免由于故障而导致工业生产过程的中断。机械师定期地检查每一台设备，在不同的位置测量设备的振动状况，判断这

台设备是否需要维修。典型的故障形式包括轴心偏移、机构松弛、轴承失效和泵动不平衡。某个化学工厂拥有超过1000多台不同的设备，范围从很小的泵到非常大的涡轮交流发电机，这些设备至今仍需要由有二十多年工作经验的专家进行诊断。通过对设备装置上的不同位置振动的测量，以及使用傅里叶分析法在三个轴向上检测出在基本转速下每一个共振所产生的能量，进行故障的判断。由于受到测量和记录过程的限制，所得到的信息非常繁杂，需要专家对这些信息进行综合研究后做出诊断。尽管手工制作的专家系统已经在一些方面得到发展，由于得到诊断方案的过程将在不同的机械装置上重复很多遍，所以也在探索运用机器学习方案。

600多个故障中的每一个故障都是由一组测量和专家的诊断结果组成，这些诊断再现了在这一领域二十年的经验。其中有一半由于不同的原因不能令人满意，而不得不丢弃，剩下的将作为训练例子投入使用。这个诊断系统的目的不是为了考查是否存在故障，而是要判断出
25 是什么故障。因此，在训练集里没有必要包括没有故障的事例。由测量得到的属性值是属于较低层的数据，必须使用一些中间的处理方法将它们增大。中间处理方法是一些基本属性的公式，这些公式是专家结合一些因果关系的领域知识所制定的。用一个归纳法公式对衍生出的属性进行运算，从而产生一组诊断规则。一开始，专家并不满意这些规则，因为他不能将这些规则与他的知识和经验相联系。对于他来说，统计的证据本身并不是一个充分的解释。为了让建立的规则令人满意，必须使用更多的背景知识。尽管产生的规则将会非常复杂的，但是专家喜欢它，因为他能够根据自己的机械知识对它们进行评估。专家对一些由第三方产生的规则和他自己使用的规则达成一致感到高兴，而且愿意从其他规则中获得新的洞察力。

从性能测试的结果可以看出，机器学到的规则较优于由专家根据以往的经验手工编制的规则，这一结果在化学工厂使用的效果中得到进一步的肯定。有趣的是，这个系统的应用不是因为它有良好的性能表现，而是因为由机器学到的规则已经得到这个领域的专家的肯定。

1.3.5 市场和销售

一些最具有活力的数据挖掘的应用是在市场和销售领域。在这个领域，各个公司都掌握着大量的精确数据记录，这些数据仅在最近才被认为拥有巨大的潜在价值。在这些应用中，预测是主要兴趣所在，用于做出决策的结构常常是完全不相干的。

我们已经提到有关客户忠诚度摇摆问题，以及找出那些可能会背叛的客户的挑战，因为这部分客户可以通过给予特殊对待而争取他们回来。银行是数据挖掘技术较早的使用者，因为数据挖掘的使用在信用度审核方面取得了成功。如今为减少储户的流失，数据挖掘被用来考查个人银行业务模型。这个模型将预示出一个银行业务的变化，甚至是生活的变化，如移居到另一个城市，这些变化可能会导致选择一个不同的银行。或者，一个通过电话管理的银行业务的客户群，当连续几小时的电话接通率较低时，这部分客户的流失率将高于平均值。
26 数据挖掘能为新推出的服务找出所适合的群体，例如一个除了在11、12月以外很少从他们的信用卡上提出现金的有利可图的、可靠的客户群，他们愿意为度个好假期而支付高昂的银行利息。在移动通信领域，移动电话公司为了留住他们的基本用户群，需要找出可能从新推出的服务中获利的行为模型，然后向这个客户群宣传新服务。为挽留所有用户而特别提供一些激励措施的代价是昂贵的。一个成功的数据挖掘能够准确的、有针对性的，对有可能产生最大利润的用户群提供特殊服务。

购物篮分析是使用关联技术在交易过程中，特别是从超市收银数据中，找出那些以成组

的形式同时出现的商品。对于许多零售商来说，这是唯一能够用于数据挖掘的销售信息来源。例如，对收银数据的自动分析将揭示一个事实：那些买啤酒的客户同时也买薯片。这是一个对超市管理人员来说也许具有重要意义的发现（尽管这是个很明显的并不需要数据挖掘技术来揭示的发现）。或者也可能找另一个事实，一些顾客通常在星期四买尿片的同时也买啤酒。这个起初令人吃惊的结论，如果我们再仔细想一下，考虑到年轻父母为了在家度周末而储藏，就变得可以理解了。这些信息能够用于多种目的：规划货架、仅对一组会同时被购买的商品中的一种商品进行打折、当一个产品单独销售时，提供与这个产品相匹配产品的赠券，等等。认知顾客个人的购买历史记录的能力可以创造出巨大的附加价值。事实上，对这个价值的追求造成折扣卡或“忠诚”卡的繁盛，让零售商无论何时都能从购买行为中鉴别出特殊的客户。从个人数据中获得的结论将比折扣的现金价值更高。对个别顾客的鉴别不但使得对其历史购买模式进行分析，而且还能精确地针对潜在的用户发送有关特殊服务的邮件。

这将带领我们进入直接的市场，另一个数据挖掘广泛应用的领域。促销行为的代价是昂贵的，它拥有低的反馈率，却能产生高额利润。任何能使一个促销邮递广告更加紧密集中，并达到给出尽量少的样品，但是获得相同或者几乎相同的反馈信息的技术都是有价值的。一些有商业价值的数据库包含着基于邮政编码的人口统计信息，邮政编码定义了邻里关联的信息，它能与现有的客户信息相关联，从中找到一个社会经济模型，进而预测出哪些客户将会转换成真正的客户。这个模型能够从一个最初的邮递广告反馈信息上，预测出有可能的未来客户。反馈信息是人们寄回的反馈卡，或者拨800电话寻求更多信息。快递公司比大型购物中心的零售商更有优势掌握单个客户的购买历史，使用数据挖掘能够从中找出那些有可能响应特殊服务的客户。有针对性的竞争比争夺大市场的竞争更经济，因为公司仅对有可能需要产品的客户提供服务，为此公司将节约大量资金。而机器学习能够帮助人们找出针对性的目标。 27

1.3.6 其他应用

现在有不胜枚举的有关机器学习的其他应用。这里我们将简单地介绍一些应用领域来说明机器学习运用领域的广泛性。

成熟的生产制造过程通常涉及调整控制参数。从天然气中分离出原油是对石油进行提炼的一个必不可少的过程，而分离过程的控制是一个比较难的工作。英国石油公司使用机器学习为设置参数建立规则。现在这个过程只需要10分钟，而以往同样的工作，专家们需要花一天多的时间完成。西屋公司（Westinghouse）在制造核燃料芯块的过程中，使用机器学习建立规则以控制生产过程。据报道因此他们每年节约超过1000万美元（1984年）。坐落在田纳西的R.R. Donnelly印刷公司，运用同样的理论控制轮转凹版印刷过程，降低了由人为因素造成的不当参数设置，人为错误的数量也由每年超过500降低到不足30。

在客户支持和服务领域，我们已经讨论过贷款审批、市场和销售应用。另一个例子是当一个客户反映电话通讯故障后，电话公司必须做出决定派什么技师解决问题。贝尔大西洋公司（Bell Atlantic）在1991年开发了用来做出这个决策的一个专家系统已经在1999年被一组由机器学习得到的规则所替代，这一举措降低了错误决策的数量，因此每年为公司节约1000多万美元。

数据挖掘也被用于许多科学领域。在生物学领域，使用机器学习能帮助人们从每一个新的染色体中识别出数千个基因。在生物医学领域，使用机器学习技术不但能够从药物的化学

属性，而且也能从它们的三维结构上分析预测出药物的作用。加速了药物开发的进程，同时降低了开发成本。在天文学领域，机器学习技术已经被用于开发一个完全自动的分类系统，用于分析那些太小、不容易被观察到的天体。在化学领域，机器学习被用来从核磁共振影像中预测出特定的有机复合物的结构。在所有这些应用中，机器学习技术所达到的性能级别（或者应该说是技能？）都能抗衡或超过人类专家。

一个需要连续监控的工作，对人类来说是一个耗时而且异常枯燥的工作，在这种情况下，自动化技术尤其受到欢迎。前面所提到的有关石油泄漏的监控是机器学习在生态方面的应用。其他的一些应用相对来说比较间接。例如，机器学习正被用来根据电视观众的以往选择和节
28 目预告，来预测出观众对电视频道的偏好。而在其他方面的一些应用甚至能挽救生命。危重病人需要进行长时间的监控，来觉察不能用生理节律和用药来解释的病人指标变化情况，等等，目的是在适当的时候发出警报。最后，一个依赖于易受攻击的计算机网络系统的世界，正日益关注网络安全性的问题，使用机器学习能够从识别非正常的操作模式的过程中探测出非法入侵。

1.4 机器学习和统计学

机器学习和统计学的区别是什么？玩世不恭者挖苦地把它看成是揭示商业利益（和骗局），将数据挖掘等同于统计加市场。实际上，试图在数据挖掘和统计之间寻求一个分界线是不现实的，因为它是一个连续的、多维的数据分析技术。其中一些技术源于标准的统计课程，另外一些更紧密地与源于计算机科学领域的机器学习相关联。从历史上来说，两方面存在一些不同的惯例。如果一定要指出一个特别不同之处，就是统计学更侧重于测试假说，而机器学习更注重于规划出一个概括的过程，作为一个在全部可能的假说中的搜索。但是这个解释过于简单化，统计学远不只是测试假说，而且许多机器学习技术也并没有包含任何搜索。

过去，一些非常相似的方案已经在机器学习和统计学上得到并行发展。其中一个是决策树归纳法。四位统计学家（Breiman等）在20世纪80年代中期出版一本书《分类和回归树》（Classification and regression trees），同时卓越的机器学习研究学家J. Ross Quinlan，在70年代和80年代初，为从样本中推导出分类树开发了一个系统。这两个独立的项目为从实例中建树产生了非常相似的方案，但是研究人员直到很晚才意识到互相的成就。第二个相似的方案产生于用于分类的最近邻方法中。机器学习研究人员对一些标准的统计技术进行了广泛的改进，使得在分类的性能得到改进的同时，提高了计算过程的效率。我们将在第4章研究决策树和最近邻方法。

如今，机器学习和统计学已经结合起来。在本书中，我们将要考察的一些技术在很大程度上融入统计的思想。从一开始，创建和提炼原始样本集时，就在数据可视化、属性的选择、
29 去除异常等等方面，运用标准的统计法。大部分学习算法在创建规则或树以及修正“过度拟合”的模型时，使用了统计的测试，过度拟合就是在产生模型过程中过分依赖于一些特定样本的细节（我们已经在图1-3劳工谈判问题的两个决策树中看到过一个过度拟合的例子）。统计测试用来验证机器学习的模型和评估机器学习算法。在研究数据挖掘实用技术时，我们将学到大量的统计知识。

1.5 用于搜索的概括

将一个学习问题可视化的方法之一是将机器学习与统计进行区分的方式。设想一个在可能的概念描述空间中与数据相匹配的搜索。尽管以概括进行搜索的想法对思索有关机器学习来说是一个强大的概念工具。但是它并不是理解这本书中所描述的特殊方案的必要部分，所以，在本节的右边有灰色线条标记，意味着“选读”。

为了明确起见，假设概念是学习的结果，由一组规则表示，像在1.2节中有关天气问题的规则（尽管其他概念描述语言也能同样做到）。假定我们要列出所有可能的规则集，然后从中寻找一些能够满足给定样本集的规则。一个大工程？是的。不可能完成？乍一看似乎如此，因为对可能产生的规则数量没有限制。但是事实上可能的规则集的数量是有限的。首先要注意，每一个单独的规则不能大于一个固定的最大值，每一个属性至多有一个条件：在表1-2中的天气数据总共包含了4个条件。因为可能的规则的数量是有限的，所以可能的规则集的数量尽管非常大，但也是有限的。然而，我们对那些拥有非常大量规则的规则集不感兴趣。实际上，我们对那些所包含的规则数量大于样本数量的规则集不感兴趣，因为很难想像每一个样本有一个以上的规则。因此，如果我们仅考虑那些小于样本数量的规则集，那么问题将得到极大的简化，尽管数量依然很大。

对于表1-3第二版本的天气问题，似乎存在一个严重的危机，因为规则中包含了数值，所以可能的概念描述的数量将趋于无限。如果它们是实数，不可能将它们一一枚举出来，甚至理论上也不可行。但是在样本中出现的数字如果用数值形式的分割点来表示，这些问题将会迎刃而解。例如，在表1-3中的温度属性，它所包含的值是：64，65，68，69，70，71，72， 30
75，80，81，83和85，共12个不同的值。存在13个可能的区间，可以为涉及温度的规则设置分割点。所以这个问题将不再是无限的。

概括的过程可以被认为是在一个庞大的，但是有限的搜索空间进行搜索。原则上说，可以枚举出所有描述，从中剔除一些不符合样本的描述，来解决问题。一个肯定的样本会去除所有不匹配的描述；而一个否定的样本则去除所有匹配的描述。每个样本剩余的描述集缩小（或与原来相同）。如果只剩下一个描述，那么它就是目标描述（目标概念）。

如果剩下多个描述，仍然可以用来对未知的事物进行分类。如果未知的事物与所有剩余的描述相匹配，它就应该被分到所匹配的目标。如果不能和其中任何一个相匹配，就把它分到目标概念以外。但是，如果仅和其中一部分相匹配，就会出现模糊。在这种情况下，如果未知事物的类已经标出，那么会导致剩余描述集缩小，因为那些错分事物的规则集会被拒绝。

1.5.1 枚举概念空间

查寻被认为是观察学习过程的一个好方法。然而，查寻空间是极大的，尽管是有限的。一般来说，枚举出所有可能的描述，并从中寻找匹配的描述是极不实际的。在天气问题中，对于每一个规则存在$4 \times 4 \times 3 \times 3 \times 2 = 288$个可能性。对于阴晴属性存在4个可能性：sunny、overcast、rainy，或者有可能根本没有参与规则的制定。同样，温度存在4个可能性，刮风和湿度分别有3个，类有2个。如果我们把规则集里的规则数量限制在14个以内（训练集里有14个样本），那就有可能产生2.7×10^{34}不同的规则集。那就需要枚举出很多描述，特别对这样一个明显很琐碎的问题。

尽管有一些方法可以使得枚举的过程更可行，但仍然存在一个严峻的问题：实际上，整

个处理过程只涵盖唯一一个可接受描述是很少见的。所以会产生两种情况，样本在处理过后，仍然存在大量的描述，或者所有的描述都被去除。第一种情况的出现是由于并没有充分理解样本，所以除“正确”的一个以外的其他描述没有被去除。实际上，人们通常想要找到一个“最佳”描述，所以有必要运用一些其他标准从剩余的描述集中选出最好的一个。第二种情况
31 的出现是因为描述语言表达不充分，不能把握住真正的概念，或者是因为样本中存在干扰。如果输入的样本由于一些属性值存在错误而导致一个“错误”的分类，或者样本的类被标错，都有可能把正确的描述从空间中去除，所以剩下的描述集就为空。如果实例中存在干扰，这种情况非常可能发生，而且必然会发生，除非是人造的数据。

另一种对作为查询的概括进行观察方法，不是将它设想成一个枚举出所有描述，并从中剔除所有不适用的描述的过程，而是作为一种在描述空间的爬山行为，根据一些预先制定的匹配标准，寻找最匹配实例集的描述。这是一种最实际的机器学习工作方法。然而，除了一些很琐碎的事例，在整个空间彻底地寻找是完全不可行的。最实际的算法应该包含引导性的查询，并且不能保证找到最优的描述。

1.5.2 偏差

把概括看成是在一个所有可能的概念空间的查询，就可以清楚地指出在机器学习系统里最重要的决策是:

- 概念描述语言
- 在空间搜索的次序
- 对于特定训练数据避免过度拟合的方法

当讨论查询的“偏差”时，通常要提到这三个属性。它们是语言偏差(language bias)、查询偏差(search bias)和避免过度拟合偏差（overfitting-avoidance)。在选择一种表达概念的语言时；为一个可以接受的描述寻找一个特殊途径时；当需要对一个复杂的概念进行简化时，都会使学习方案产生偏差。

语言偏差

对于语言偏差来说，最重要的问题是概念描述语言是否具有普遍性，或者它是否在能够被学到的概念上加了约束条件。如果考虑一个包含所有可能性的样本集，那么概念就是将这个集合划分为多个子集的分界线。在天气例子中，如果要枚举出所有可能的天气条件，那么概念玩就是一个所有可能的天气条件的子集。一个“普遍性的”语言应该能够表示出每一个可能的样本子集。实际上，可能的样本集普遍很大，从这方面说，这只是一个理论上的设想，而不能用于实际。

如果概念描述语言允许包括逻辑或，也就是析取，那么任何一个子集都能表示。如果描述语言是基于规则的，那使用分开的规则就能够达到分离的目的。例如，一个可能的概念描述仅仅是枚举样本:

32
```
If outlook = overcast and temperature = hot and humidity = high
   and windy = false then play = yes
If outlook = rainy and temperature = mild and humidity = high
   and windy = false then play = yes
If outlook = rainy and temperature = cool and humidity = normal
   and windy = false then play = yes
If outlook = overcast and temperature = cool and humidity = normal
```

```
    and windy = true then play = yes
  ...
  If none of the above then play = no
```

这不是一个具有启发性的概念描述：只是简单地记录了已经被观察到的一些肯定的样本，并且假设剩下的都是否定的样本。每一个肯定的样本被赋予一个自身的规则，概念是规则以逻辑或的形式构成。或者，可以设想为每一个否定的样本制定一些单独的规则，同样也会得到乏味的概念。在这两种方法里，概念描述并没有表示出任何概括，只是简单地记录原始数据。

然而，如果不允许使用逻辑或的原则，一些可能的概念（样本集）将不可能被表达出来。从这个意义上说，一个机器学习方案也许不会轻易达到一个好的表现。

其他语言偏差是因为在特殊领域所使用的知识而产生的。例如，它或许是一些属性值的组合，而这些组合可能永远不会发生。当一个属性意味着另一个属性的时候，这种情况将会发生。在1.2节，在为大豆问题寻找规则的时候我们曾经看到一个相关的例子。当然，考虑包含冗余或不可能的属性值组合的概念是没有意义的。领域知识能够用于削减查询空间。知识就是力量，即使是一点知识也能起很大作用，甚至一个小小的线索都能极大地减少搜索空间。

搜索偏差

在实际的数据挖掘问题中，存在一些可选的概念描述，它们都与数据相匹配。问题是要根据一些标准（通常是简单的标准）从中找出“最佳”的一个。我们使用统计学的术语拟合(fit)，意味着寻找一个与数据合理拟合的最佳描述。但是，要在整个空间进行搜索并保证所找到的描述是最好的一个，通常从计算上是不可行的。所以搜索过程是启发式的，并且不能做出有关最终结果是最优的保证。这为“偏差”的产生创造了空间：不同的搜索引导方式以不同的方式在搜索中产生偏差。

例如，一个学习方案或许为规则的产生采纳一个“贪心”搜索，通过尝试在每一个阶段找出最好的规则，然后把它加入规则集。然而，最好的一对规则并不是单独找出的两个最佳规则的叠加。或者当构造一个决策树的时候，早先做出的基于一个特定属性的分割，或许在考虑如何在这个节点下发展树时会被认为是错误的决定。使用限定范围搜索（beam search）解决这些问题，方法是不产生一个不能改变的约束方案，而是并行地寻求一个由多个动态替换方案组成的集合，替换方案的个数就是“束宽”(beam width)。这将使学习算法变得非常复杂，但是将有可能避免与贪心查询相关联的短视。当然，如果束宽不够大，短视有可能发生。还有一些更复杂的查询策略能够帮助解决这个问题。 33

一个更通用的和更高层次的查询偏差需要调查查询是由一般性的描述开始，再对它进行提炼，还是由一个特殊的样本出发，然后对它进行推广。前者称为从一般到具体的搜索偏差，而后者是从具体到一般的搜索偏差。许多学习算法采用前一种策略，由一个空的决策树，或是一个非常一般的规则开始，对它进行具体化处理，使它与样本拟合。然而，从另一个方向对它进行研究也是非常可行的。基于实例的方法从一个特定的样本出发，观察它是如何被一般化后覆盖同一个类里的相邻的样本。

避免过度拟合偏差

避免过度拟合偏差是另一种形式的搜索偏差。但是因为它涉及一个比较特殊的问题，所以我们将它加以区别对待。前面已经讨论过逻辑或的问题，如果允许包含逻辑或，在总结数据的时候就会存在没有用的概念描述，然而如果禁止使用逻辑或，一些概念将无法得到。通

常解决这个问题的方法是从一个最简单的概念描述出发，逐渐将它复杂化：由简到繁的顺序。这是为了迎合简单的概念描述，从而使查询产生偏差。

采用由简到繁的查询开始，在所找到足够复杂的概念描述的时候停止，是一个避免过度拟合的好方法。有些时候称为正向修剪（forward pruning）或前修剪（prepruning），因为一些描述将在变得复杂以前就被修剪掉。采用反向修剪（backward pruning）或称后修剪（post-pruning）也是可行的。首先，我们找出一个与数据拟合很好的描述，然后将它向回修剪成一个简单的同样也与数据拟合的描述。看上去这是多余的一步，其实不然。通常为找出一个简单的理论，首先需要找到一个复杂的，然后对它进行简化。正向和反向修剪都可避免过度拟合偏差。

总之，作为查询的概括是思考有关学习问题的一个好方法，而实际操作过程中，偏差是
34 唯一使得它可行的方法。不同的学习算法对应不同的概念描述空间，并用不同的偏差来查询。有趣的是：不同的描述语言和偏差在解决某些问题上表现突出，而在另一些问题上却差强人意。正如每一个老师都知道的，绝对好的学习方法是不存在的。

1.6 数据挖掘和道德

数据挖掘中的数据的使用，特别是有关人的数据，隐含着严肃的道德问题。数据挖掘技术的运用者必须意识到围绕在特殊应用上的道德问题，从而负责地使用它们。

当使用对象是人时，数据挖掘频繁地用于区别待遇：谁得到贷款，谁得到特殊待遇，等等。某些形式的区别对待：如种族、性别、宗教等等，不仅是不道德的，而且是非法的。然而，情况是复杂的：每一件事取决于应用。在医疗诊断中使用性别和种族的信息当然是道德的，但是在研究贷款支付行为上使用同样信息却是不道德的。甚至在丢弃一些敏感信息以后，创建的模型仍然存在歧视的风险，因为模型可能依赖于一些变量，这些变量是种族或性别特征的替代品。例如，人们居住的一些地区总是和一些特定的种族特征相关联，所以在一个数据挖掘研究中使用邮政编码就会使模型中存在种族歧视的危险，尽管一些种族信息已经明确地从数据中删除。

在人们决定提供个人信息之前，需要知道如何使用这些信息，及使用这些信息的目的。采取什么措施保障这些信息的机密性和完整性，提供或者保留这些信息会有什么样的后果，以及他们应该拥有一些纠正信息的权利。无论何时在收集这些信息的时候，相关人员都应该被告知这些事宜。不仅是以一种有法律效应的打印件的形式，而是直接用朴实的语言，使他们能够理解。

数据挖掘技术的潜在使用意味着，数据库里的数据在使用方面会得到扩展，也许会远远超出在开始收集数据时所作的设想。这将产生一个严肃的问题，即，必须明确收集数据的条件，以及使用目的。数据的拥有者能把所收集的数据用于收集之初所告知的使用目的以外的其他方面吗？对于明确地收集的个人数据当然不能。但是总的来说存在一些复杂的情况。

从数据挖掘中能够发现一些令人惊奇的东西。例如，据报道法国的一个主要消费团体
35 已经发现，拥有红色汽车的人们往往不按期支付他们的汽车贷款。这种发现的作用是什么？它基于什么信息？这些信息是在什么条件下收集的？以一种什么样的途径来有道德地使用它？显而易见，保险公司在业务中根据一些教条来区别对待人群，如，年轻的男性支

付更多的汽车保险费，这些教条并没有完全基于统计学的关联，它们也没有包含有关这个世界的普通常识。是否以上的发现能够说明选红色的车的人的一些情况，或者是否它应该被作为不相关的部分加以抛弃，这些都需要根据人们对世界的理解而不是纯粹统计的标准来做出判断。

当你得到数据的时候，你需要问谁可以使用它，收集它的目的是什么，以及什么样的结论是可以合法地从中得到。道德的尺度对数据挖掘实践者提出了严峻的问题。在处理数据的过程中，考虑使用一些社会规范是非常有必要的。一些标准也许已经被发展几十年或几个世纪，但是信息专家也许并不清楚。例如，你知道吗？在图书馆的社区里，读者的隐私被理所当然地视为一种权利，应该得到保护。如果你打电话给大学图书馆，询问某某书被谁借走了，他们将不会告诉你。这将保护学生免受由于屈服一个发怒的教授所施加的压力而交出书，而这本书是他为最近提交的一份奖学金申请所急需的。这也将禁止对大学道德委员会主席怀疑的娱乐阅读品位进行深入探询。而创建数字图书馆的人或许并没有意识到这些敏感问题，为了向读者推荐新书，而用数据挖掘系统分析数据和比较个人的阅读习惯，甚至可能会把结果卖给出版商。

除了数据的使用要符合社会标准外，还必须使用符合逻辑的和科学的标准来分析从中得出的结论。如果你确实得到一些结论（如拥有红色车的人的信用风险比较大），你需要对这些结论附加说明，这种说明要用非纯粹的统计报告论据来支持。关键是在整个过程中，数据挖掘仅仅是一个工具。人们得到结果后，还要结合其他知识，然后才能决定采取什么行为。

数据挖掘还推出了其他问题，当考虑在数据挖掘中使用到哪些社会资源时，它就真正成为一个政治的问题。前面我们已经提到数据挖掘在分析购物篮上的应用，通过分析超市的收银记录，洞察人们所购买的东西之间的关联。所得到的信息有什么用途？超市经理是否应该把啤酒和薯片放在一起，来方便客户？或者将它们远远地分开，让客户稍感不便，从而尽量 [36] 延长他们在超市的时间从而增大他们买未列入计划商品的可能性？超市经理是否应该把最昂贵的、利润最高的尿片移到靠近啤酒的位置，来增加对被折磨的父亲们的高利润商品的销售，还要在边上加上高级婴儿用品？

当然，任何一个使用高级技术的人都应该深入思考他们在做什么。如果数据被定义为记录的事实，那么信息就是基于数据的模型集，或者是期望。可以把知识定义为期望集的累积，把智慧定义为附加在知识上的价值。尽管我们不在这里研究它，但是这些问题是值得思考的。

正如我们在这一章的一开始所看到的，在这本书中阐述的技术也许会用于做一些生活中最复杂、最迫切的决定。数据挖掘是一个需要我们严肃对待的技术。

1.7 补充读物

为了避免打断正文的阅读，所有文献将收集在每一章的最后一节。第一个补充读物部分列出了在第1章中所涉及的相关的论文、书和其他资源。在这一章的开始所提到的人工受精的研究项目是牛津大学计算实验室承担的，奶牛筛选的研究项目是由新西兰怀卡托大学计算机科学系承担的。

天气问题的例子来源于Quinlan（1986年），已经广泛用来解释机器学习方案。从绪论到1.2节所提到的天气问题的主体部分出自Blake等（1998年）。隐形眼镜的例子出自Cendrowska（1987年），我们将在第4章学习由他提出的PRISM规则学习法。鸢尾花数据集出自一篇早期的

在统计推论方面的经典论文（Fisher 1936年）。劳工谈判的数据来自《集体谈判评论》（Collective Bargaining Review），一个加拿大工会的出版物，由工业关系信息服务发行（BLI 1988年）。Michalski和Chilausky在1980年首先记述了大豆问题。

在1.3节提到的一些应用出自一篇优秀的论文，给出了丰富的机器学习应用和规则归纳法（Langley和Simon 1995年）；另一个有关领域应用的案例出自一本机器学习杂志（Kohavi和Provost 1998年）。Michie（1989年）详细阐述了信贷公司应用；浮油探测的应用来自Kubat 等
37 （1998年）；电力负载预测工作的应用源于Jabbour 等（1988年）；电子机械设备预防性维护应用出自Saitta和Neri（1998年）。在1.3节提到的一些其他更完整的描述（包括节约的美元的数值，有关的文献参考）出自 Alberta Ingenuity Centre 机器学习的网站（www.aicml.cs.ualberta.ca/About/success1.htm——译者注）和欧洲一个有关机器学习的网络MLnet（www.mlnet.org/resources/showcase.htm——译者注）。

在Quinlan的一系列论文和一本书中（1993年），提到他的方案与由Breiman等在1984年独立出版的书《分类和回归树》中介绍的方案相似（第1.4节）。

第一本有关数据挖掘的书出版于1991年（Piatetsky-Shapiro和Frawley 1991），它收集了在20世纪80年代末的一个关于从数据库里发现知识的研讨会上发表的论文。另一本书出自1994年的一个研讨会（Fayyad 等1996年）。还有一些是以商业为主的数据挖掘方面的书，它们主要关注于实际方面的应用：如何把位于所使用的方法下的比较表面化的技术运用到实际当中。它们是有价值的应用和启示的来源。例如，来自Syllogic的Adriaans和Zantige（1996年），一个欧洲的系统和数据库的咨询服务公司，很早以前就对数据挖掘做过介绍。来自宾夕法尼亚州的一个专门研究数据仓库和数据挖掘公司的Berry和Linoff（1997年），为市场、销售和客户支持给出了一个出色的基于样本的数据挖掘技术回顾。五个来自IBM跨国实验室的研究人员（Cabena等1998年）用许多真实世界的应用例子概括了数据挖掘的过程。Dhar和Stein（1997年）给出了有关数据挖掘在商业方面的前景，包括对许多技术的大致的，通俗的回顾。为提供数据挖掘软件公司工作的Groth（1998年）给出了一个简洁的数据挖掘的绪论，然后对数据挖掘软件产品进行了完整的广泛的回顾。这本书包括一张他们公司产品演示版本的光盘。Weiss和Indurkhya（1998年）为他们称为“大数据”的数据做出预测，广泛研究了不同的统计技术。Han和Kamber（2001年）从数据库的角度涉及数据挖掘，着重于从大型的相关数据库中发现知识。最后，这个领域里受到广泛尊敬的一个国际作家团体Hand等，在2001年撰写一本各个学科在数据挖掘应用方面的书。

另外，有关机器学习的书往往使用学术的文体，比较适合于大学课程而不能作为实际应用的指导。1997年Mitchell出版了一本出色的书，它涵盖了大量的机器学习技术，包括了一些本书未提及的基因算法和增强学习法。Langley在1996年写了一篇很好的文章。尽管上面提到的Quinlan（1993年）的书中集中讨论一个特殊的学习算法C4.5，我们将在第4章和第6章中详
38 细介绍，对于许多机器学习的问题和技术来说，它仍是一个很好的启蒙。从统计的角度来看Hastie等在2001年出版的书，绝对是一本卓越的机器学习的书。它从理论上来指导研究，并且采用了很好的例证。

模式识别是一个与机器学习紧密相关的主题，它运用了许多相同的技术。2001年Duda 等出版的有关模式识别（Duda和Hart，1973年）的第二版是一本优秀和成功的书。Ripley（1996年）和Bishop（1995年）阐述了用于模式识别的神经网络。用神经网络进行数据挖掘是

Bigus（1996年）所著的书里的一个科目，Bigus为IBM工作，这本书介绍了由他开发的IBM神经网络应用产品。

如今很多人对支持矢量机器学习的技术感兴趣。见第6章。Cristianini和Shawe-Taylor（2000年）给出了一个很好的介绍，细述了有关应用的附加算法，内核和解决方案，并用模式找出生物信息，文本分析和图像分析领域里的问题（Shawe-Taylor和Cristianini 2004年）。Schölkopf和Smola（2002年）是两个年轻的研究人员，他们在这个迅速发展的机器学习的分支上做他们的博士论文。他们在论文中详细介绍了支持矢量机器和有关的核心方案。 39

第2章 输入：概念、实例和属性

在深入研究机器学习方案如何运作以前，需要了解可以采取哪些不同形式的输入。下一章将介绍会产生哪些不同形式的输出。与所有软件系统一样，了解输入输出分别是什么远比理解软件如何工作更为重要，机器学习也不例外。

机器学习的输入采用概念、实例和属性的形式。能够被学习的事物称为一个概念描述。机器学习中的概念，乍看上去就和学习一样，很难给出精确的定义，这里也不打算纠缠它是什么和不是什么的哲学问题。从某种意义上说，学习过程的结果即，一个概念的描述，正是我们所要发现的，是可以理解的（intelligible），能够被理解、商讨和辩论，并且是可以操作的（operational）：能够被运用到实际的样本上。下一节将介绍不同机器学习问题之间的一些
41 差异，这些差异在实际数据挖掘中非常具体，也非常重要。

机器学习中，信息以实例集的形式呈现给学习者。正如第1章所述，每一个实例都是一个将要被学习的、独立的概念样本。当然，许多情况下原始数据并不能由一些单一的、独立的实例表示。也许应该把背景知识作为输入的一部分考虑进来。原始数据可能会以一大块的形式出现，并不可以被拆成单一的实例。原始数据也可能是一个序列，像一个时间序列，如果将它剪成数段后将会失去意义。然而，本书是有关数据挖掘的简单而实用的方法，着重点在于研究那些能够以独立样本的形式出现的信息。

每一个实例由测量实例不同方面的一些属性值所定性。属性存在许多不同的类型，尽管典型的数据挖掘方案只处理数值属性和名词性属性，或称范畴属性。

最后，我们将考察为数据挖掘而进行数据输入准备的问题，并且介绍一个与本书配套的Java软件所使用的简单格式。它用一个文本文件表示输入信息。

2.1 概念

在数据挖掘应用领域里存在四种完全不同的学习方式。*分类学习*（classification learning）是用一个已分类的样本集来表示学习方案，并希望从这个样本集中学习对未来样本进行分类的方法。*关联学习*（association learning）：寻找任何特性之间的关联，不仅仅是为了预测一个特定的类值。*聚类*（clustering）：寻找能够组合在一起的样本，并依此分组。*数值预测*（numeric prediction）：预测出的结论不是一个离散类而是一个数值量。不管采用什么方式进行学习，这里将被学习的东西称为*概念*，由学习方案产生的输出就是*概念描述*。

第1章里的大部分例子属于分类问题。天气问题（表1-2和表1-3）是一组日期的集合，并且对每一天标注了是否适合进行体育活动的判断结果。问题是要学习如何用是否玩对新的一天进行分类。存在于隐形眼镜数据（表1-1）里的问题是要学习如何向一个新的病人推荐一副适合的眼镜，或者把问题再明确一点，因为这套数据展示了所有属性值的组合，需要学习对所给数据进行总结的方法。从鸢尾花数据（表1-4）中要学习如何根据花萼的长度和宽度，以及花瓣的长度和宽度推测出新鸢尾花属于setosa、versicolor或virginica中的哪一种。对于劳工
42 谈判数据（表1-6），要根据第一、二、三年的工作时间长度、工资增长幅度以及生活费调整

等等，判断一个新的合同能否被接受。

分类学习有时又称为有指导（supervised）的学习，因为从某种意义上来说，学习方案是在指导下操作的，这里所说的指导即每一个训练样本都有一个明确的结论。如是否玩的判断、推荐的隐形眼镜类型、鸢尾花的品种、劳工合同的可接受性。这些结论称为样本的*类*。把学到的概念描述在一个独立的数据测试集上进行测试，已知这个测试集的分类，但是对于机器它是未知的，以此来判断分类学习是否成功。在测试数据上的成功率是对所学到的概念的一个客观评价。在许多实际数据挖掘应用中，衡量数据挖掘成功的标准是以人类使用者对所学到的概念描述：规则、决策树，或其他任何结果的接受程度所决定的，是一个主观的评价。

第1章的大部分例子同样可以用于关联学习，关联学习中没有指出特定的类。问题是如何从数据中找出"有趣的"结构。1.2节已经给出了一些天气数据的关联规则。关联规则和分类规则在两个方面存在不同：关联规则可以"预测"任何一个属性，不只是类，也可以一次预测一个以上的属性值。正因为如此，关联规则的数量要远远多于分类规则的数量，其中的问题是要避免被过多的关联规则所困扰。所以，通常要为关联规则制定一个能够适用的最小样本数量，如80%的数据集，并且还要大于一个特定的最小正确率，如正确率为95%。尽管如此，仍然会产生大量的规则，因此必须手工验证它们是否具有意义。关联规则通常仅包含非数值的属性，一般不会从鸢尾花数据集中寻找关联规则。

当样本不存在一个特定的类时，可以采用聚类的方法将那些看上去会自然落在一起的样本集合在一起。表2-1是一个省略了鸢尾花类型的数据版本。150个实例很可能自然地落入3个与鸢尾花类型相对应的聚类内。其中的挑战是要找出这些聚类，并把实例分配到各个聚类上，还要能够将新的实例分配到相应的聚类上。有可能一个或多个鸢尾花品种自然地分成多个子聚类，这时产生的自然聚类数量将超过3个。聚类的成功与否通常以所得到的结论对人类使用者是否有用来主观地衡量。它常伴随着第二步分类学习，所学到的规则将给出如何把新的实例安置到相应聚类里的可以理解的描述。 43

表2-1 聚类问题的鸢尾花数据

	花萼长(cm)	花萼宽(cm)	花瓣长(cm)	花瓣宽(cm)
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
5	5.0	3.6	1.4	0.2
…				
51	7.0	3.2	4.7	1.4
52	6.4	3.2	4.5	1.5
53	6.9	3.1	4.9	1.5
54	5.5	2.3	4.0	1.3
55	6.5	2.8	4.6	1.5
…				
101	6.3	3.3	6.0	2.5
102	5.8	2.7	5.1	1.9
103	7.1	3.0	5.9	2.1
104	6.3	2.9	5.6	1.8
105	6.5	3.0	5.8	2.2
…				

数值预测是分类学习的一种变体，数值预测的结论是一个数值，而不是一个分类。CPU性能问题就是一个预测数值的例子。表2-2展示了另一个例子，在这个版本的天气数据里，预测值不是玩或者不玩，而是玩的时间（分钟）。对于数值预测问题，以及其他的机器学习，所学到的描述性的结构往往比对新实例的预测值更能引起人们的兴趣。这个结构指出了哪些是重要的属性，以及它们与数值结论存在什么样的关系。

表2-2　数值类的天气数据

	阴　晴	温　度	湿　度	刮　风	玩的时间 (min.)
	sunny	85	85	false	5
	sunny	80	90	true	0
	overcast	83	86	false	55
	rainy	70	96	false	40
	rainy	68	80	false	65
	rainy	65	70	true	45
	overcast	64	65	true	60
	sunny	72	95	false	0
	sunny	69	70	false	70
	rainy	75	80	false	45
	sunny	75	70	true	50
	overcast	72	90	true	55
	overcast	81	75	false	75
44	rainy	71	91	true	10

2.2　样本

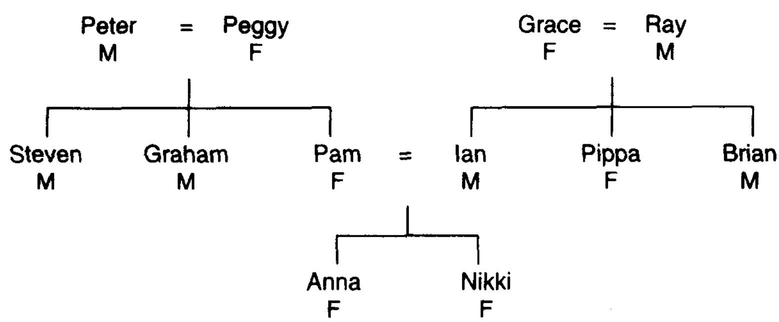
机器学习方案的输入是一个实例集。这些实例由机器学习方案进行分类、关联或聚类。尽管到现在为止，它们被称为样本，但是从现在开始将用更专业的术语：实例，来表示输入。每一个实例都是一个被用来学习的单一、独立的概念样本。每个实例由一组预先定义的属性值来表示。上一章讨论过的所有数据集（天气、隐形眼镜、鸢尾花和劳工谈判问题）中的实例都是这样产生的。每一个数据集都可以表示成一个实例与属性的矩阵，用数据库的术语说这是单一关系的数据，或一个平面文件（flat file）。

用一个独立的实例集来表示输入数据是到目前为止用于实际数据挖掘中最普遍的方法。然而这种阐明问题的方法存在一些局限性，下面对局限性存在的原因进行解释。实例之间通常存在某种关系，并不真正是彼此分离、独立的。例如，从一个给出的家族树上学习姐妹的概念。想像你自己的家族树，并把你所有的家族成员（和他们的性别）分别放置在各个节点上。这棵树，以及成员的名单和对应的他们是否存在姐妹关系的描述就是学习过程的输入。

图2-1显示了一个家族树的一部分，树下面的两个定义姐妹关系的表格所用的方法稍微有点不同。第三列里的yes意味着第二列的家族成员和第一列里的家族成员存在姐妹关系。（这是一个建立在给定样本上的随意判断）

首先需要注意的是左边那张表的第三列存在很多no。因为每列有12个成员，所以总共有12 × 12 = 144对成员，其中大部分的成员对并不存在姐妹关系。右边的表给出了同样的信息，但只记录了肯定的实例，并假设剩余的都是否定的实例。仅明确指出肯定样本且采用一个不
45 变的假设（剩下的都是否定的样本）的做法称为闭合世界假定（closed world assumption）。

这是理论研究中常用的假设，但是在解决实际问题时它并不实用，闭合世界意味着包含了所有事件，而在实际中很少存在闭合的世界。



第一人	第二人	是姐妹吗？
Peter	Peggy	no
Peter	Steven	no
…	……	
Steven	Peter	no
Steven	Graham	no
Steven	Pam	yes
Steven	Grace	no
…	……	
Ian	Pippa	yes
…	……	
Anna	Nikki	yes
…	…..	
Nikki	Anna	yes

第一人	第二人	是姐妹吗？
Steven	Pam	yes
Graham	Pam	yes
Ian	Pippa	yes
Brian	Pippa	yes
Anna	Nikki	yes
Nikki	Anna	yes
其余		no

图2-1　一个家族树和两种表示姐妹关系的方法

图2-1中的两张表都不能离开家族树单独使用。这个树同样能够用表格形式来表示，表2-3显示了以表格形式表达的该树的部分内容。现在这个问题可以用两种关系来表达。但是这些表不包含独立的实例集，因为判断姐妹关系所依据的列（姓名、父母1和父母2）的值需要参考家族树关系的行。表2-4是把两张表合并成一张表后产生出一个单独的实例集。 46

表2-3　由一张表表示家族树

姓名	性别	父母1	父母2
Peter	male	?	?
Peggy	female	?	?
Steven	male	Peter	Peggy
Graham	male	Peter	Peggy
Pam	female	Peter	Peggy
Ian	male	Grace	Ray
…			

表2-4 在一张表中表示姐妹关系

第一人				第二人				是姐妹吗?
姓名	性别	父母1	父母2	姓名	性别	父母1	父母2	
Steven	male	Peter	Peggy	Pam	female	Peter	Peggy	yes
Graham	male	Peter	Peggy	Pam	female	Peter	Peggy	yes
Ian	male	Grace	Ray	Pippa	female	Grace	Ray	yes
Brian	male	Grace	Ray	Pippa	female	Grace	Ray	yes
Anna	female	Parn	Ian	Nikki	female	Pam	Ian	yes
Nikki	female	Pam	Ian	Anna	female	Pam	Ian	yes
其余								no

现在已经成功地把原始关系问题转换成实例的形式，每个实例都是一个单一、独立的概念样本，这个样本被用来学习。当然，实例并没有真正独立，在表的不同行之间存在许多关系，但是如果从姐妹关系的概念角度考虑，它们是独立的。许多机器学习方法在处理这类数据时仍然存在问题，这些问题将在第3.6节中讨论，但是现在至少已经把数据重新转化为正确的形式。一个姐妹关系的简单规则是：

```
If second person's gender = female
   and first person's parent1 = second person's parent1
   then sister-of = yes
```

这个例子说明了如何从一棵树上获取不同节点间的关系，并且转换成一个独立的实例集。用数据库的术语说，获取了两个关系，并且把它们结合成一个。这是一个平整的处理过程，技术上称为反向规格化（denormalization）。对一些存在（有限的）关系的（有限的）集来说，这个过程是可以实现的。

47 表2-4所示的结构能够用来表示两个家庭成员间的任何亲属关系：祖孙关系、孙子与祖父的兄弟之间的关系或其他关系。要表示出更多家庭成员之间的关系就需要一张更大的表。如果家庭成员的总数没有事先明确，将对关系的表示造成很大的困难。假如要学习核心家庭（nuclear family）（父母和他们的孩子）的概念，所包含的人的总数取决于人口最多的核心家庭，尽管可以猜测一个合理的最大数字（10或20），但是真正的核心家庭成员数量只能通过扫描整棵树来找出。然而，如果有一个存在有限关系的有限集，至少在理论上，可以产生一个新的记录了家庭成员之间的所有组合的“超级关系”，无论有多少家庭成员，这个“超级关系”足以表示出成员之间的任何关系。但是，从计算和存储的代价来说这种方法都是不可行的。

反向规格化存在的另一个问题是在一些数据上产生了显而易见的规律性，而这些规律性完全是虚假的，实际上它们仅是原始数据库结构的再现。例如，设想一个超市数据库里存在一个顾客和他所买商品之间的关系，一个商品和供应商之间的关系，一个供应商和供应商地址之间的关系。经过反向规格化后将产生一个平面文件，每一个实例将包含顾客、商品、供应商和供应商地址。一个在数据库中寻找结构的数据库挖掘工具也许会得出一个事实：买啤酒的顾客也买薯片，对超市经理来说这是一个重大的发现。然而，它也许也会产生另一个事实：供应商的名字能准确地“预测”出供应商的地址，这个“发现”根本不可能引起超市经理的任何兴趣。这个伪装成一个从平面文件中得到的重大发现的事实，已经由原始数据库结构明确显现出来。

尽管任何真正的输入实例集必须是有限的，但是许多抽象的计算问题包含了一些非有限

的关系。像祖辈关系的概念包括一个贯穿树的任意长度的路径，虽然人类的家族树或许是有限的（尽管大得惊人），但是很多人造问题产生的数据确实是无限的。这听起来也许很深奥，但是这种现象在类似于列表处理及逻辑编程等领域却很常见，且在一个称为归纳逻辑编程的机器学习分支领域里有过探讨。在可能的实例数量是无限的情况下，计算机科学家通常使用递归的方法进行处理。例如:

```
If person1 is a parent of person2
   then person1 is an ancestor of person2
If person1 is a parent of person2
   and person2 is an ancestor of person3
   then person1 is an ancestor of person3
```

不管两个人的关系有多远，都能用这个简单的祖辈关系的递归定义来表示。归纳逻辑编程技术能够从一个像表2-5所示的有限实例集里学习递归规则。 48

表2-5 用表描述另一种关系

第一人				第二人				是长辈吗?
姓名	性别	父母1	父母2	姓名	性别	父母1	父母2	
Peter	male	?	?	Steven	male	Peter	Peggy	yes
Peter	male	?	?	Pam	female	Peter	Peggy	yes
Peter	male	?	?	Anna	female	Pam	Ian	yes
Peter	male	?	?	Nikki	female	Pam	Ian	yes
Pam	female	Peter	Peggy	Nikki	female	Pam	Ian	yes
Grace	female	?	?	Ian	male	Grace	Ray	yes
Grace	female	?	?	Nikki	female	Pam	Ian	yes
			其他样本集					yes
			其余					no

递归技术的一个真正缺陷是不善于处理干扰数据，除了一些小的人造数据集外，对其他数据集来说，它的处理速度往往太慢以致不能用。本书没有涉及这个技术，但Bergadano和Gunetti（1996年）曾提出了一个综合处理法。

总之，一个数据挖掘方案的输入是一个独立概念的实例表，并将用于学习。正因为如此，有人不屑地建议应该称它为文件挖掘（file mining）而不是数据库挖掘（database mining）。关系数据比一个平面文件更复杂。虽然常常耗费大量的空间，但是一个有限关系的有限集总是能被重新转换成单个表。此外，反向规格化会从数据中产生虚假的规则性，所以有必要在实施一个机器学习方案前检查是否存在虚假事实。最后，递归学习规则具有可以处理无限的概念的潜在能力，但是这超出了本书的范围。

2.3 属性

每个单一、独立的实例是由一组固定的和预先定义的特征或属性值作为输入提供给机器学习的。实例是如天气、隐形眼镜、鸢尾花和CPU性能问题的表的行；属性是列。（劳工谈判数据是一个例外，因为空间的原因用列表示实例，用行表示属性。）

在实际的数据挖掘所考虑的问题中，一个固定特征集的使用是一种限制条件。如果不同实例有不同的特征会怎样？例如，假设实例是交通工具，车轮的数量是一个可以用于许多车辆的特征，但是不能用于船只；桅杆的根数是船只的特征，但不适用于陆地车辆。一种标准 49

的工作方法是把每一个可能的特征作为一个属性，并使用一个“无关值”的标记指出对于一个特定的案例哪个属性不适用。当一个属性的存在（如：配偶的名字）取决于另一个属性值时（已婚或未婚）就会出现类似的情况。

一个特定实例的一个属性值是属性所对应部分的一个测量值。数值量和名词性量之间存在明显的差异。数值属性有时也称为连续属性，它是测量到的实数或整数值。需要注意的是，从数学的观点上说，整数值在数学意义上当然是不连续的，这里滥用了连续这个术语。名词性属性是从一个预先定义的有限的可能值的集合中取值，有时候也称为范畴属性。但是也存在其他可能性，在统计的文章中经常介绍“测量标准”，如名词性值、有序值、区间值和比率值。

名词性值是一些独特的符号。这些值作为标签或者名字使用，所以称它们为名词性(nominal)。这个词出自拉丁文name。例如，在天气数据中，outlook（阴晴）的属性值是：sunny、overcast和rainy。这三个值之间没有隐含任何关系，没有先后次序或距离测量。把值进行相加或者相乘，或是比较它们的大小也是没有意义的。使用这类属性的规则只能测试相等或不等。如

```
outlook: sunny    → no
         overcast → yes
         rainy    → yes
```

有序值是那些有可能进行排序的范畴值。尽管值间有排序的可能，但是绝不存在距离。例如，在天气数据里，气温（temperature）的属性值是：炎热（hot）、温和（mild）和凉爽（cool）。它们是有序的。为了方便起见是否可以把它们看成：

hot>mild>cool或者hot<mild<cool

只要保持连贯性，两者皆可。重要的是要把温和放置在其他两个值之间。尽管在两个值之间进行比较是有意义的，但是将它们相加或者相减都没有意义。hot和mild之间的差异不能和mild和cool之间的差异进行比较。使用这类属性的规则可能包括一个比较。如下所示：

```
temperature = hot → no
50 temperature < hot → yes
```

注意名词性的量和有序的量之间的差异并不总是直接明了的。实际上，以上使用的这个有序量的样本，如outlook，就并不十分清楚，也许还会出现分歧意见，认为三个值之间确实存在序列。多云（overcast）可以被认为是介于sunny和rainy之间的值，它是天气由好转坏的一个过渡阶段。

区间值不但是有序的而且还可以用固定和相等的单位进行度量。温度就是一个很好的例子，它用度表示（如：华氏），而不是用cool、mild和hot非数值的刻度来暗示。讨论两个温度的差异是很明确的工作，如46度和48度，也可以与其他两个温度之间的差异进行比较，如22度和24度。另一个例子是日期。可以讨论1939年和1945年之间的差异（6年），甚至可以计算1939年和1945年的平均值（1942年），然而将1939年与1945年相加（3884），或者将1939年乘以3（5817），都没有任何意义。因为作为开始点的0年完全是臆想出来的，在历史上它已经更改很多次了。（孩子们有时候会疑惑公元前300年在当时是如何称呼的。）

比率值的测量方法内在定义了一个零点。例如，当测量一个物体到另一个物体的距离时，物体到它自身的距离形成一个自然的零值。比率值通常是实数，所以可以进行任何数学运算。当然将距离乘以3也是合理的、两个距离相乘将得到面积。

然而，问题是一个“固有”定义的零点是否基于我们的科学知识?科学知识是和文化相联

系的。例如，华氏并没有最低温度的限制，它的刻度是一个区间。但是现在我们把温度看成基于绝对零的一个比率值。用年来计算时间是基于由文化定义的零，如公元元年，它不是一个比率刻度，而是开始于宇宙大爆炸的年份。在谈论钱的零点时，人们通常会谈到某一物品要比另一种物品贵一倍，但对于我们中间那些不断将信用卡刷爆的人来说，这种谈论就不是很有意义了。

许多实际的数据挖掘系统只采用四种测量标准中的两种：名词性值和有序值。名词性属性有时称为范畴的、可枚举的或离散的属性。枚举是计算机科学里的一个标准术语，用来表示一个范畴的数据类型，但严格的定义应该是它与自然数字有一对一的对应关系，其中隐含了一个顺序的关系，但是在机器学习中没有隐含顺序关系。离散也含有顺序的关系，因为通常需要离散一个连续的数量值。有序的属性经常称为数值的或连续的属性，但是不存在数学连贯性的暗示。名词性值的一个特例是二分值，它只有两个值，通常设 51
计成如天气数据里所见到的true和false，或者yes和no的形式。这类属性有时也称为布尔属性。

机器学习系统可以使用许多有关属性的其他种类信息。例如，空间上的考虑能够用来把搜索限制在表示或者比较那些在空间上正确的数据。循环的顺序会影响到多种测试方案的制定。例如，在时间的一个日常概念中，对一个属性是天（day）的测试可能涉及明天、前天、下一个工作日、下周的同一天。部分排序是一般或者具体的关系，在实际中频繁发生。这种信息通常作为元数据被提及，元数据是关于数据的数据。然而，当今的一些用于数据挖掘的实际方案很少有能力将元数据考虑进来，不过这种能力会在未来得到迅速发展。（第8章将讨论有关内容。）

2.4 输入准备

在整个数据挖掘过程中，为数据挖掘研究所做的数据输入准备工作常常要花费大量的精力。本书并不打算真正讨论数据准备问题，只是指出其中包含的一些问题，从中认识到它的复杂性。接着将讨论一个特殊的文件输入格式，属性相关文件格式（ARFF格式），它是第二部分Java程序包使用的一个输入格式。然后阐述在数据集转换成ARFF格式过程中将会产生的问题，以及需要注意的一些简单实际的要点。实验表明真实数据的数据质量低得令人失望，所以被称为数据清理的一个仔细检查数据的过程是数据挖掘的重要步骤。

2.4.1 数据收集

在着手开始研究数据挖掘问题之前，首先需要把所有数据集中成一个实例集。在讨论家族树时已经解释了将关系数据进行反向规格化的原因。尽管它揭示了数据集中的基本问题，但是这种独立的非人造的样本，并没有真正反映出在实际是怎样的一个过程。而真正的商业应用需要从不同部门收集数据。例如，在营销研究中，需要从销售部门，顾客账单部门和顾客服务部门收集数据。

将不同的数据源进行整合是一项具有挑战性的工作，虽然不存在深奥的原则性问题，但是处理起来却很棘手。因为不同的部门也许使用不同的记录形式、不同的习惯、不同的时间段、不同的数据集合度、不同的主键和不同的错误形式。所以数据必须集中、整合和清理。大型数据整合的思想称为数据仓库（data warehousing），数据仓库提供了一个访问成组数据的 52

接口，它超越了部门的界限。旧数据在数据仓库中以商业决策借用的方式发布。把数据转移到数据仓库将证实一个事实：将各个部门在日常工作中产生的一些部门级信息聚集起来后，会产生出巨大的战略价值。显而易见，数据仓库的存在是数据挖掘工作的一个非常有用的先决条件，如果没有数据仓库，在为数据挖掘做数据准备工作时，必须采用数据仓库所包含的许多数据处理步骤。

通常一个数据仓库所含的数据并不都是必需的，有时需要越过部门把数据和相关的问题联系起来分析。例如，在上一章中讲述的在电力负载预测里考虑天气数据，以及在市场和销售应用中考虑人口统计学的数据。这些数据有时称为重叠数据（overlay data），它通常不是由一个组织收集的，而是明显与数据挖掘问题有关。当然重叠数据也必须清理，并且要与其他已经收集到的数据进行整合。

数据整合的另一个实际问题是什么程度的数据整合是合理的。当牛奶农场主决定出售哪些牛时，牛的产奶量记录是一个重要的决定因素。每条牛的产奶量由一个自动的挤奶机器每天记录两次，所以需要将数据进行合并。同样，当电话公司研究客户行为时，有一些电话访问的原始数据是不会用到的，必须把这些数据整合到客户层。数据是按月使用还是按季度使用，或者推迟几个月或几个季度？选择正确的数据类型和数据整合的程度通常关系着数据挖掘的成功与否。

因为数据整合中涉及很多问题，不能期望在一开始就取得成功。这就是为什么数据集中、清理、整合和一般的数据准备工作需要花费很长时间。

2.4.2 ARFF格式

现在介绍ARFF文件，它是一个由独立的、无序的实例组成的数据集的标准表示方法，该表示方法不涉及实例之间的关系。

图2-2是表1-3中天气数据的一个ARFF文件，这个版本的一些属性是数值属性。由%开始的行是注释行。在文件开始处紧接着文件注释的是关系的名称（`weather`）和一组属性的定义（`outlook`，`temperature`、`humidity`，`windy`，`play?`）。在名词性属性后面括号里的是一组名词性值。如果名词性值内包含空格，必须加引号。数值属性跟随着一个关键字：
53 数值的（`numeric`）。

尽管天气问题需要从其他属性值中预测出类值：`play?`，但是在数据文件中类属性与其他属性并没有任何区别。ARFF文件格式只给出了一个数据集，并没有指出将要预测哪些属性。这意味着可以在同样的文件上考察每一个属性究竟能否从其他属性预测出，或用同样的文件来寻找关联规则和聚类。

在属性定义下以`@data`开始的行，是数据集中实例数据开始的标志。每一行表示一个实例，属性值按照属性的顺序排列，并用逗号隔开。如果有残缺值，将由问号表示（在这个数据集里没有残缺值）。在ARFF文件里的属性规范可以用来检查数据，确保所有属性值是有效的，数据检查工作可以由读入ARFF文件的程序自动完成。

ARFF格式除了有像天气数据中存在的名词性属性和数值属性外，还有其他两种属性形式：字符串属性和日期属性。字符串属性的值是文本。如果需要定义一个叫description的字符串属性，应该用如下形式定义：

```
@attribute description string
```

```
% ARFF file for the weather data with some numeric features
%
@relation weather

@attribute outlook { sunny, overcast, rainy }
@attribute temperature numeric
@attribute humidity numeric
@attribute windy { true, false }
@attribute play? { yes, no }

@data
%
% 14 instances
%
sunny, 85, 85, false, no
sunny, 80, 90, true, no
overcast, 83, 86, false, yes
rainy, 70, 96, false, yes
rainy, 68, 80, false, yes
rainy, 65, 70, true, no
overcast, 64, 65, true, yes
sunny, 72, 95, false, no
sunny, 69, 70, false, yes
rainy, 75, 80, false, yes
sunny, 75, 70, true, yes
overcast, 72, 90, true, yes
overcast, 81, 75, false, yes
rainy, 71, 91, true, no
```

图2-2　天气数据的ARFF文件

在一个实例数据中，可以用引号包括任意的字符串（如果在字符串里包含引号，要在每个引号前加上反斜线“\”）。字符串内在地存储在一个字符串表中，并由它们在表里的地址表示。因此含有相同字符的两个字符串将拥有同样的值。

字符串属性包含的值可以非常长，甚至可以是一个文件。为了能使用字符串属性进行文本挖掘，有必要对它们进行处理。例如，也许可以将一个字符串属性转换成大量的数值属性，字符串中的每一个单词对应一个数字，这个数字是单词在字符串中出现的次数。此种转换方式将在7.3节中讨论。

日期属性是一个特殊形式的字符串，用以下方式定义：

```
@attribute today date
```

（表示一个叫today的属性）。Weka是本书第二部分要讨论的机器学习软件。它使用ISO-8601标准组合日期和时间，格式为：yyyy-MM-dd-THH:mm:ss，用4位数字表示年，分别用2位数字表示月和天，字母T以后各用2位数字表示小时、分钟和秒。[⊖]在文件的数据部分，日期用相应的日期和时间的字符串表示，例如“2004-04-03T12:00:00”。尽管它们被表示成字符串，但是在文件读入时，日期将被转换成数值形式。日期也可以内部转换成多种不同格式，所以在数据文件里可以使用绝对时间戳和一些转换方法，以形成一天中的时间或者一周里的天，从而便于考察阶段性的行为。

⊖ Weka在属性定义部分有一个将日期属性定义成包含一个特殊字符串的不同格式的机制。

54~55 2.4.3 稀疏数据

有时大部分实例的很多属性值是0。例如，记录顾客购物情况的购物篮数据，不管购物清单有多大，顾客购买的商品都只占超市所提供的一小部分。购物篮数据记录了顾客所购买的每种商品的数量，除此以外的几乎所有库存商品的数量都是0。数据文件可以看成一个由行和列分别表示顾客和所存储商品的矩阵，这个矩阵是稀疏矩阵，几乎所有的项都是0。另一个例子出现在文本挖掘中，这里的实例是文档。而矩阵行和列分别表示文档和单词，用数字表示一个特定的单词在文章中出现的次数。由于大部分文章的词汇量并不大，所以很多项也是0。

明确地表示出一个稀疏矩阵的每一项并不实际。下面按序表示每个属性值：

```
0, 26, 0,  0, 0, 0, 63, 0, 0, 0, "class A"
0,  0, 0, 42, 0, 0,  0, 0, 0, 0, "class B"
```

作为代替的另一种表示方法是将非0值属性用它的属性位置和值明确标出。如：

```
{1 26, 6 63, 10 "class A"}
{3 42, 10 "class B"}
```

收集每一个非0值属性的索引号（索引从0开始）和属性值，并将每一个实例包含在大括号里。在ARFF文件格式里，稀疏数据文件包含相同的@relation和@attribute标签，紧接着是一个@data行，但是数据部分的表示方法不同，由以上所示的在大括号里用明确表述的方法表示。注意省略的值都是0，它们并不是“残缺”值！如果存在一个未知值，它必须用一个问号明确地表示出来。

2.4.4 属性类型

ARFF文件格式允许两种基本数据类型：名词性值和数量值。字符串和日期属性实际上也分别是名词性值和数量值，尽管字符串在使用以前通常需要转换成数值形式，如字向量。但是对两种基本类型的诠释方法取决于所使用的机器学习方案。例如，很多机器学习方案把数值属性作为有序的刻度处理，并且仅在数值间进行小于和大于的比较。然而，另一些将它们作为比率值处理，并且使用距离计算。所以将数据用于数据挖掘以前，要理解机器学习方法的工作原理。

如果机器学习法处理数值属性是用比率值测出的，将会引出一个标准化的问题。属性值
56 通常被标准化成一个固定的范围，如从0到1，把所有的值除以所有出现的值中最大的一个，或者先减去一个最小值，然后除以最大和最小值之差。另一个标准化技术是计算属性值的统计平均值和标准差，把每一个值减去统计平均值后除以标准差。这个过程称为将一个统计变量标准化，并且所得到的是平均值是0，标准差是1的值的集合。

一些机器学习方案，如各种基于实例的不同的机器学习和回归方法只能处理比率值，因为它们根据属性的值计算两个实例之间的“距离”。如果实际的值是有序的，必须定义一个数值距离公式。一种处理方法是使用两层的距离：1表示两个值不同，0表示相同。任何名词性值都能够作为数值用距离公式处理。但它是一个有点粗糙的技术，掩盖了实例间真正不同的程度。另一种可行的方法是为每一个名词性属性建立综合的二值属性。有关内容将在6.5节使用树进行数值预测时涉及。

有时名词性值和数值之间存在真正的映射关系。例如，邮政编码指定的区域可以用地理坐标来表示；电话号码的前几位也有相同功能，它与所在的区域相关。学生证号码的前两位

数也许是学生入学的年份。

实际的数据集普遍存在名词性值作为整数编码的情形。例如，一个整数形式的标识符也许用来作为一个属性的代码，如零件号码，但是这些整数值不能用于小于或大于的比较。如果是这样，明确指出属性是名词性值而不是数值是非常重要的。

把一个有序值作为名词性值处理是可行的。实际上，许多机器学习方案只处理名词性值。例如，在隐形眼镜问题里，年龄属性就是作为名词性值来对待的，所产生的一部分规则如下：

```
If age = young and astigmatic = no and
   tear production rate = normal then recommendation = soft
If age = pre-presbyopic and astigmatic = no and
   tear production rate = normal then recommendation = soft
```

事实上年龄（特别是这种形式的年龄）是一个真正的有序值，可以用以下方式表示：

```
young < pre-presbyopic < presbyopic
```

如果将它作为有序值对待，那么上面两条规则就可以合并成一个： 57

```
If age ≤ pre-presbyopic and astigmatic = no and
   tear production rate = normal then recommendation = soft
```

这是一个更紧凑，更满意的表示相同含义的方法。

2.4.5 残缺值

实际中遇到的很多数据集包含残缺值，例如表1-6中的劳工谈判数据。残缺值通常是指超出正常范围，可能会在正常值是正数的位置出现一个负数（如：-1），或在一个正常情况下不可能出现0值的位置出现0。对于名词性属性，残缺值可能会由空格或横线表示。如果需要对不同形式的残缺值加以区别（如：未知、未记录、无关值），可以用不同的负整数表示（-1，-2等）。

必须仔细研究数据中残缺值的意义。残缺值的出现有多种原因。例如，测量设备出现故障，在数据收集过程中改变了试验方法，整理几个相似但不相同的数据集。被访问者在访问中也许会拒绝回答某些问题，如年龄或收入。在考古研究领域，一个样本，如一个头盖骨被损坏了，导致某些参数不能测出。在生物学研究领域，所有参数在测量以前，植物或动物就已经死了。这些情况的出现对纳入考虑之中的样本意味着什么？也许头盖骨的损坏有某种意义，或者仅仅是随机的事件？植物已死去这个事实有意义或是相反？

很多机器学习的方案隐含地假设：一个实例的某个属性值残缺并没有特别意义，这个值只是未知而已。然而，这个值为什么残缺也许会有一个很好的理由，可能是基于所了解的信息而做出的决策，不执行某些特定测试。如果是这样，这其中提供的关于实例的信息要比仅仅了解有残缺值这个事实多得多。如果是这样，也许将属性的可能值记录为“未测试”更为妥当，也可将此作为数据集中的另一个属性。上面的例子说明，只有熟悉数据的人才能做出一个明智的判断：一个特定值的残缺是否存在一些特别的意义，是否应该将它作为一个一般的残缺值进行处理。当然，如果存在几种类型的残缺值，那就意味着出现了异常状况，需要调查具体原因。

如果残缺值意味着一个操作员曾经决定不进行一个特定的测量，那么它将传达出较之这个值是未知的这个事实更多的信息。例如，人们在分析医学数据库时已经注意到，一般情况下病情的诊断是按照医生决定所做的测试的结果，但是有时侯，医生不需要知道测试结果就 58

可以做出诊断。在这种情况下，带有某些残缺值的记录是做一个完整诊断所需的全部，那些实际值可以被完全忽略。

2.4.6 不正确的值

仔细检查数据挖掘文件从中找出不良的属性和属性值是非常重要的工作。数据挖掘中使用的数据并不是为了数据挖掘而收集的。在最初收集数据时，数据的某些方面可能并不重要，所以留下空白或没有被检查。由于这样不会对收集数据的初衷造成任何影响，所以不用更正它。然而，当这个数据库用于数据挖掘时，错误和省略的部分立刻变得相当重要。例如，银行并不真正需要知道客户的年龄，所以它们的数据库中也许会存在许多残缺或不正确的（有关年龄的）值。但是在由数据挖掘得到的规则中，年龄也许会成为一个非常重要的特性。

数据集的印刷错误显然会造成不正确的值。通常表现为名词性属性的值被拼错，这将为名词性属性制造一个额外的值。或者不是拼错，而是一个同义词，如百事和百事可乐。很明显，一个事先定义的格式，如ARFF格式的优势在于可以检查数据文件以保证数据内部的连贯性。然而，在原数据文件中出现的错误通常会经过转换过程保存到用于数据挖掘的文件里。因此每一个属性所拥有的可能值的列表都应该仔细检查。

印刷或测量在数值上造成的错误通常会导致超出范畴的值，可以通过一次取一个变量进行作图的方法检查错误。错误的值往往会远离一个由其余的值构成的模式。当然，有时候要找出错误值是困难的，尤其是在一个不熟悉的知识领域里。

重复的数据是另一种错误源。如果数据文件中的一些实例是重复的，很多机器学习工具将会产生不同的结果，因为重复将对结果产生很多影响。

人们在向数据库输入个人数据时常常会故意制造一些错误。他们也许会在拼写他们的地名时做一些小的改动，试图确认他们的信息是否会被出售给广告商，而使他们收到大量的垃圾邮件。如果以前一些人的保险申请曾经被拒绝过，也许他们会在再次申请保险时修正他们名字的拼写。严格的计算机化的数据输入系统通常强制性地要求输入一些信息。例如，一个外国人在美国租车时，计算机坚持要他在美国的邮政编码。可是他来自其他国家，根本没有美国的邮政编码。万般无奈之下，操作人员建议他使用租车公司的邮政编码。如果这是很普
59 遍的做法，那么在以后的数据挖掘中将出现一群客户的地址和租车公司在同一区域。同样，超市出纳员有时会在顾客不能提供购物卡时使用他们自己的购物卡，为了让顾客得到折扣，或者是为了在出纳员的账户上增加积分。只有深入了解有关的背景知识，才能够解释如上所示的系统性的数据错误。

最后，数据也存在有效期。随着周围情况的变化，数据也会发生变化。例如，邮件单里的姓名、地址、电话号码等项就经常会发生变化。所以在数据挖掘里需要考虑用于挖掘的数据是否依旧有效。

2.4.7 了解数据

在数据挖掘中有必要加强对数据的理解。一些可以显示名词性属性值的柱状分布图，和数值属性值的分布图（也许将实例进行排序，或绘图）的简单工具都有助于理解数据。用图形的方式将数据可视化能够方便地鉴别出界外值，是一个很好的表示数据文件中错误的方法，也为非正常情况的编码提供了很大的便利。如用9999表示一个残缺的年或 - 1kg表示一个残缺

的重量、人们并没有提供这些表示法的说明。还需要与领域的专家商量来解释反常的、残缺的值、以及那些用整数表示范畴而不是真正数量值的重要性，等等。将属性值两两进行坐标投影，或者将各个属性与对应的类值进行坐标投影都将有助于对数据的理解。

数据清理是一个费时费力的过程，却是成功的数据挖掘所绝对必要的。人们经常放弃一些大型的数据集，就是因为他们没有可能完全核对数据。取而代之，可以抽取一些实例仔细研究，从中会得到惊人的发现。所以花一些时间来审视数据是值得的。

2.5 补充读物

Pyle （1999年）为数据挖掘提供了一个详尽的数据准备的指导。现在许多人对数据仓库和它所呈现出来的问题很感兴趣。据我们所知Kimball（1996年）对此类问题做的阐述是最好的。Cabena（1998年）等认为数据准备的工作量在一个数据挖掘应用中占到60%，他们也谈到其中所包含的一些问题的工作量。

Bergadano和Gunetti（1996年）对处理有限和无限关系的归纳逻辑编程进行了研究。Stevens（1946年）引入了有关属性的不同“测量等级” 的概念，并且在统计包（如SPSS，Nie等，1970年）的相关手册里进行了详尽的阐述。 60

第3章 输出：知识表达

本书提供的大部分数据挖掘技术能轻易地产生出一些可理解的描述，而这些描述是关于数据中的结构模式的。在了解这些数据挖掘技术是如何工作以前，首先必须知道数据中的结构模式是如何表达的。机器学习所能发现的模式有许多不同的表达方式，每一种方式就是一种推断数据输出结构的技术。一旦理解了输出结构的表示方法，就向数据输出结构是如何产生的理解前进了一大步。

在第1章中给出了很多数据挖掘的例子，这些例子的输出采用的形式是决策树和分类规则，这是许多机器学习法所采用的基本知识表达形式。对一个决策树或者一个规则的集合而言，知识是一个名不副实的词，在这里用这个词并不意味着我们想要暗示这些结构胜过浮现在我们脑海里的真正的知识，只是需要用一个词描绘由机器学习方法产生出的结构。在一些更加复杂的规则中允许使用例外，可以表示不同实例的各个属性值之间的关系。一些特殊形式的
61 决策树能用来预测数值。基于实例的表达方法则着重于研究实例本身，而不需要像规则一样，分析实例的属性值。最后，还有一些机器学习方法会产生出一些实例的聚类。这些不同的知识表达方法是与第2章中介绍的不同机器学习问题相对应的。

3.1 决策表

表示机器学习输出结构的最简单、最基本的方法是采用和输入同样的形式——决策表（decision table）。例如，表1-2是一个天气数据的决策表，只需要从中寻找一些适合的条件来确定是否运动。为了更简化些，建立一个决策表也许需要涉及属性选择的问题。例如，如果温度属性和决策无关，形成更小的、扼要的表就是一个很好的决策表。当然关键问题是要确定去除哪些属性而不会影响最终的决策。

3.2 决策树

一个“分治法”用于从独立实例集学习的方法，自然引伸出一个称为决策树（decision tree）的表达形式。我们已经看到了一些决策树的例子，如隐形眼镜（图1-2）和劳工谈判（图1-3）数据集。一个决策树上的节点包含了对某个特定属性的测试。一般来说，在一个节点上的测试是将一个属性值与一个常量进行比较。然而，有一些树节点上的测试是在两个属性之间进行比较，或者使用一个包含一个或多个属性的函数公式进行比较。叶节点对所有到达叶子的实例给出一个分类，或是一组分类，或是一个包括了所有可能分类的概率分布。当对一个未知实例进行分类时，将根据在各个连续节点上对未知实例的属性值测试的结果，自上而下地从树上寻找出一条路径，当实例到达叶子时，实例的分类就是叶子所标注的类。

如果在一个节点上测试的属性是名词性属性，那么在这个节点之下产生的分支的个数就是这个名词性属性所有可能属性值的个数。在这种情形下，因为每个可能的名词性属性值对应一个分支，所以相同的名词性属性将不会在以后的建树过程中再次被测试。而有些时候，名词性属性值被分成两个子集，那么就只能产生两个分支，（实例的分配）取决于属性值所在

的子集。在这种情况下，一个名词性属性也许会在一条路径上被不止一次地测试。

如果属性是数值属性，那么在一个节点上的测试通常是判断这个数值是否大于或者小于 62
某一个事先定义的常量，给出一个二叉分裂（two-way split）。或者也可能使用三叉分裂，将会出现多个不同的可能性。如果把残缺值也作为一个独立的属性值看待，那么将会产生出第三个分支。对于一个整数的数值属性的另一种处理方法，是用小于、等于和大于实行三叉分裂。而对于实数值的数值属性来说，等于的操作并没有实际意义，所以在实数上的测试应该是用一个区间而不是一个常量，同样也可以用落在区间以下、区间内和区间以上的判断实行三叉分裂。一个数值属性通常要在给出的任何一条从树根到叶子的路径上被测试多次，每一次测试都会采用一个不同的常量。6.1节将详细讨论处理数值属性的方法。

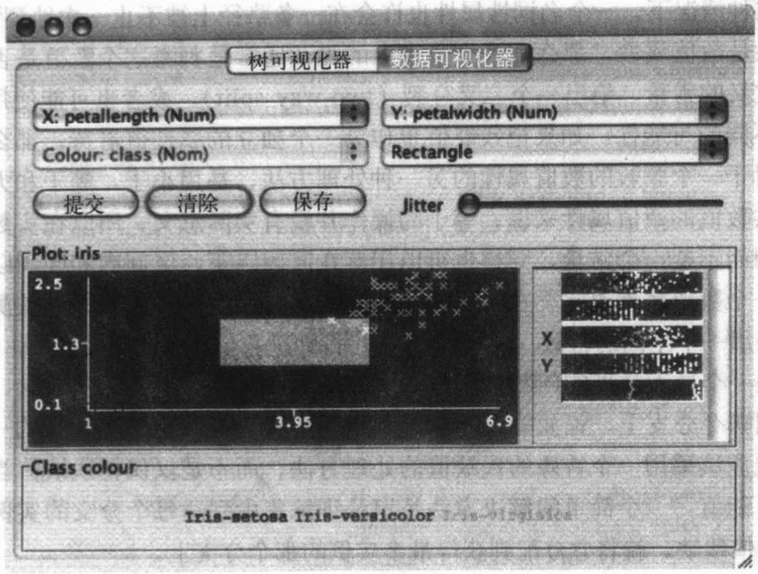
残缺值是一个显而易见的问题。当在一个节点上所测试的属性值残缺时，就不能确定应该将它分配到哪个分支上。正如第2.4节所讨论的，有时将残缺值作为属性的一个独立的值来处理。否则就应该采用一个特殊的残缺值的处理方法，而不是仅仅把残缺值当作属性可能拥有的另一个可能值。一个简单的解决方法是记录训练集中到达每个分支的实例数量，如果一个测试实例的值残缺，就将它分配到获得最多实例的那个分支上。

一个更成熟的解决方法是将实例分裂成几个部分，然后分别将它们分配到下面的每个分支上，并且由此向下，直到到达子树所包含的叶子。分裂过程采用0到1之间的权值来完成，一个分支所拥有的权值与到达这个分支的训练实例成比例，所有权值之和为1。一个加权的实例也许在较低的节点上会再次被分裂。最后，实例的不同部分将分别到达叶节点，到达叶节点后的实例分类的决策，必须由渗透到叶节点的权值重新组合后产生。第6.1节将介绍这部分内容。

对一个数据集进行手工建树是具有启发性的，甚至是很有趣的。为了有效地建树，需要有一种观察数据的好方法，因为通过观察可以判断出哪个属性有可能成为用于测试的最佳属性，以及应该采用哪种适当的测试方法。在第二部分介绍的Weka探索者里有一个用户分类器(User Classifier)，用户可以使用这个工具以交互的方式创建一个决策树。它根据用户选择的两个属性绘出一个数据散布图。当找到一对能够很好地区别实例类别的属性时，用户可以在散布图上围绕适合的数据点画出一个多边形将数据一分为二。

例如，图3-1a是用户正在一个有3个类的数据集（鸢尾花数据集）上操作，并且已经找出两个属性：花瓣长度属性和花瓣宽度属性，这两个属性能够很好地将数据按类进行分离。手工画出的一个长方形分离出其中的一个类（鸢尾花 versicolor）。然后，用户可以切换到决策 63
树视图（图3-1b）来观察。左边的叶节点主要包含了一种类型的鸢尾花（鸢尾花versicolor，仅有两个virginica被错分在这里）；右边的叶节点主要包括另外两种类型的鸢尾花（鸢尾花setosa和virginica，有两个鸢尾花versicolor被错分在这里）。用户也许选择右边的叶子进一步分析，用另一个矩形，或者基于一个不同的属性对，再继续将数据分裂（尽管图3-1a所示的两个属性看上去是很好的选择）。

第10.2节介绍了如何使用Weka用户分类器。大多数人兴致勃勃地用这个工具做出前几个决策以后，很快就失去了兴趣，一个非常有用的方法是选用一个机器学习法，让机器学习法在决策树的节点上做数据分离的工作。从手工创建决策树的过程中能够体验到，对不同属性组合分裂（数据）的评估是一项十分乏味辛苦的工作。



a) 建立一个包含花瓣长度和花瓣宽度的矩形测试



b) 得到（未完成的）决策树

图3-1 用交互的方式创建一个决策树

3.3 分类规则

分类规则（classification rules）是取代决策树的一种普遍使用的方法，前面已经介绍了一些例子：天气（第1.2节）、隐形眼镜（第1.2节）、鸢尾花（第1.2节）和大豆（第1.2节）数据集。一个规则的前提或者先决条件是一系列的测试，就像在决策树节点上的测试，而结论则给出适合与规则所覆盖实例的一个或多个分类，或者是给出实例在所有类上的概率分布。通常，先决

条件是用逻辑与(AND)的方式组合在一起，如果使用一个规则，那么必须要通过所有的测试。然而，在一些规则的表达公式中，先决条件通常是一些普通的逻辑表达式，而不是一些简单的逻辑与的组合。我们通常认为逻辑或（OR）能有效地将独立的规则组合在一起，如果其中的任何一个规则适用于这个实例，那么将规则结论得到的类（或概率分布）赋予这个实例。但是，当几个规则得出不同的结论时，就会引出矛盾的问题。我们将在下面很快涉及到这个问题。

从一个决策树上直接地读出一组规则是容易的。每一片叶子可以产生出一条规则。规则的先决条件包含了从根到叶子路径上所有节点的条件，规则的结论是叶子上标注的类。这个过程能产生明确的规则，它们执行的次序是无关的。但是，通常从决策树上直接读出的规则的复杂度远远超出所需，所以，为了去除一些冗余的测试，常常需要对从决策树上得到的规则进行修剪。

因为决策树不易表示出隐含在一个规则集里的不同规则间的逻辑或（disjunction）关系，所以将一个普通的规则集合转换成一个决策树并不是十分直截了当的。当规则拥有相同的结构，却拥有不同属性时就是反映这个问题的一个很好的例子，例如： 64~65

```
If a and b then x
If c and d then x
```

有必要打破这种对称形式并且为根节点选择一个测试。例如，如果选择a，那么第二条规则必须在树上重复两次，如图3-2所示。称为重复子树问题 (replicated subtree problem)。

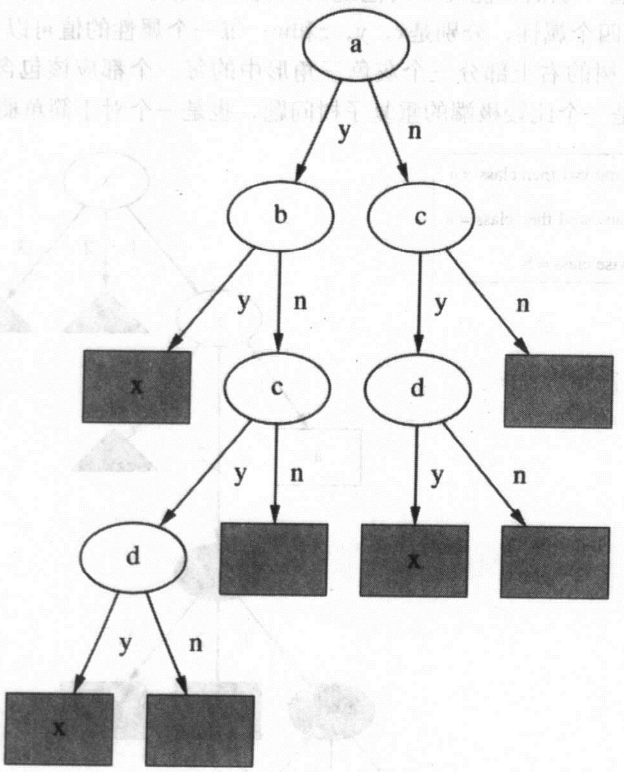


图3-2 一个简单逻辑或关系的决策树

重复子树问题是非常重要的问题，再来看几个例子。图3-3左边的图显示了一个异或（exclusive-or）函数，如果x=1或y=1，但是不能同时等于1，输出就是a。将它转变成树时，必

须先根据一个属性进行分离，产生一个如中间部分所示的结构。相对而言，规则能忠实地反
66 映出有关属性的真正的对称问题，如右边所示。

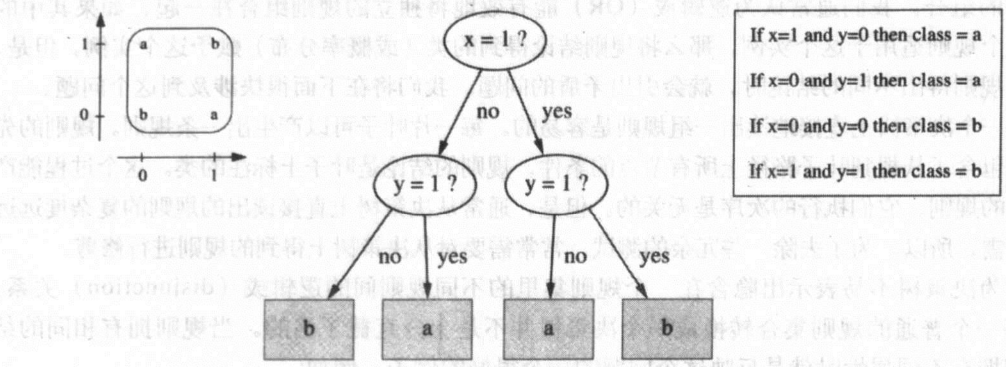


图3-3 异或问题

在这个例子中的规则并不比决策树简洁。实际上，它们只是用一种明显的方式从决策树上读取规则。但是在其他情况下，规则比决策树更加紧凑，特别是当有可能获得一个“缺省”规则时，（这个“缺省”规则）能覆盖其他规则未说明的情形。例如，在图3-4中找出规则的效应，这个规则里有四个属性，分别是x，y，z和w，每一个属性的值可以是1，2或3，右边是由规则得到的树。在树的右上部分三个灰色三角形中的每一个都应该包含一个完整的三层子树（灰色部分），这是一个比较极端的重复子树问题，也是一个对于简单概念的复杂描述。

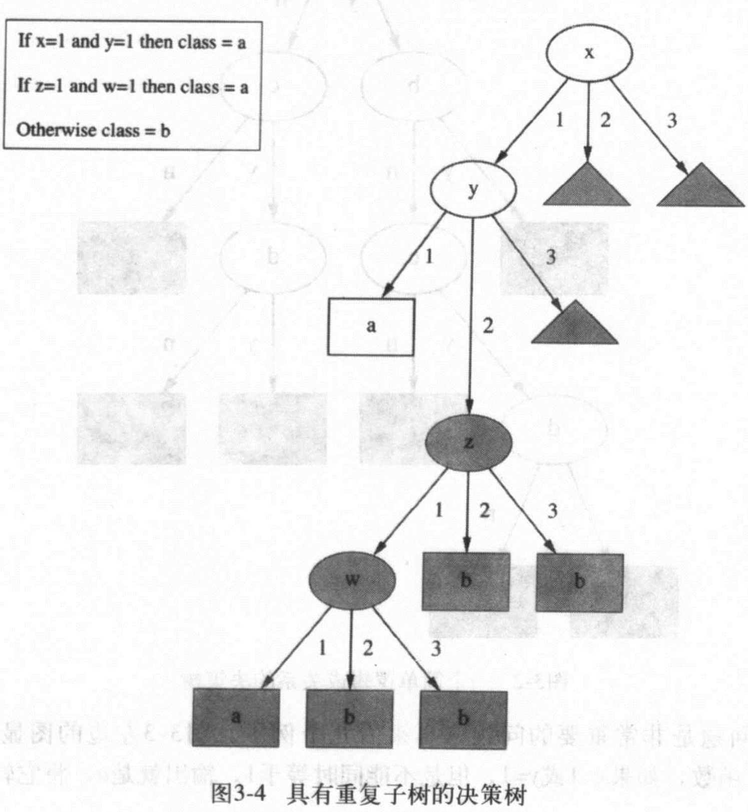


图3-4 具有重复子树的决策树

规则受欢迎的一个原因是每一条规则似乎都表示一个独立的知识“金块”。新的规则可以添加到一个已有的规则集中，却不会扰乱已经存在的规则，而向一个决策树结构添加新的规则后，则需要重新改造整个树的结构。然而，这种规则的独立性是一种错觉，因为它忽略了如何执行规则集的问题。前面已经讨论过（第1.2节），如果规则意味着像一个“决策列”按照先后次序来解释，那么单独地取出其中的一部分规则也许是不正确的。但是，如果解释的次序并不重要，那么当不同的规则在相同的实例上产生不同的结论时，就不清楚应该如何进行妥善处理。当规则是直接从决策树上读出时，这种情况并不会出现，因为存在于规则结构里的冗余，将阻止任何在解释过程中出现的模糊情况。但是，当规则是由其他方法产生时，确实会产生模棱两可的情况。

如果一个规则集对一个特定的样本给出了多个分类，一种解决方法是不给出任何结论。另一种方法是统计每一条规则在训练数据上适用的频率，选择频率最高的一条规则所对应的结论作为这个样本的分类。这些策略会导致产生完全不同的结论。当规则不能对一个实例进行分类时，就发生了另一个不同的问题。决策树或者从决策树上读出的规则不会出现这个问题。然而这种情况很容易发生在普通的规则集上。一种处理方法是对这种样本不进行分类，另一种方法是选择出现频率最高的类作为缺省类。同样，这些策略也有可能产生完全不同的结论。单独的规则是简单的，而规则的集合看上去似乎也很简单，但是如果给出的规则集并没有附上额外的信息，仍然不清楚如何对它进行解释。

一个特别简单明了的情况出现在当规则产生布尔值的类时（如yes和no），而且只采用那些仅产生一个结果（如yes）的规则。假设一个特定的实例不是类yes，一定是类no，这是一个闭合世界的假定。这样一来，规则之间就不会产生任何冲突，在规则解释的过程中也不会出现模棱两可的情况，任何解释的方法都将给出相同的结果。这种规则集可以写成一个逻辑表达式，称为析取范式(disjunctive normal form)，即表达为对逻辑与（AND）条件进行逻辑或（OR）运算的形式。 67~68

正是这个简单的特例，使得人们产生设想：规则是很容易处理的。因为这里的每一条规则确实被当作一个新的、独立的信息块来操作，采用一种简单的办法为逻辑或做贡献。不幸的是，这种方法只适用于结论是布尔值的情况，并且要求一个闭合世界的假设，而这两个限制条件在实际情况中都是不现实的。在存在多个类的情况下，由机器学习算法产生的规则必然会产生有序的规则集，这将牺牲模块化的可能性，因为规则执行的次序是非常重要的。

3.4 关联规则

关联规则（association rule）能够预测任何属性，不仅仅是类，所以关联规则也能预测属性的组合，除此以外关联规则与分类规则并没有什么不同。关联规则在使用的时候不像分类规则那样被组合成一个规则集来使用。不同的关联规则揭示出数据集的不同规律，通常用来预测不同的事物。

因为从一个很小的数据集上能够产生出很多不同的关联规则，所以只局限于研究那些能够应用在实例数量比较大，并且能在实例上获得较高正确率的关联规则。一个关联规则的覆盖量（coverage）是关联规则能够正确预测的实例数量，通常称为支持（support）。正确率（accuracy）通常称为置信度（confidence），是将正确预测的实例数量表示为它在关联规则应用所涉及的全部实例中占据的比例。例如，对于规则：

```
If temperature = cool then humidity = normal
```

覆盖量是那些温度属性是凉爽，湿度属性是normal的天数（在表1-2的数据中有4天），正确率是湿度属性为normal的天数在温度为凉爽的天数中所占的比例(这个例子的正确率是100%)。通常需要明确最小覆盖量和正确率，只寻找那些覆盖量和正确率至少达到预定最小值的关联规则。例如在天气数据中，有58条覆盖量和正确率分别至少是2和95%的规则。(将覆盖量转换为一个相对于实例总数的百分比的形式也许会更为方便。)

可以预测多个结果的关联规则在解释的时候必须小心处理。例如，表1-2所示的天气数据
69 中的一条关联规则如下:

```
If windy = false and play = no then outlook = sunny
                                and humidity = high
```

它并不仅仅是以下两个独立规则的简写形式:

```
If windy = false and play = no then outlook = sunny
If windy = false and play = no then humidity = high
```

前一条规则确实暗示了下两条规则能达到最小覆盖量和正确率，但是除此以外，它还暗示了更多的信息。前一条规则意味着没有风、不能玩与晴天、湿度大的样本个数，至少达到了指定的最小覆盖量。同时，它也意味着这种天气的天数在没有风、不能玩的天数中所占的比例、至少达到了指定的最小正确率。它还隐含了下面的规则:

```
If humidity = high and windy = false and play = no
   then outlook = sunny
```

因为这条规则与原先的规则有相同的覆盖量，并且它的正确率一定至少和原先规则相同，这是由于湿度大、没有风、不能玩的天数必然少于没有风、不能玩的天数，所以正确率会提高。

如上所示，特定关联规则之间存在关系：一些规则隐含另一些规则。当有多条规则相关联时，要减少所产生的规则的数量，合理的做法是给用户提供最重要的一条规则。上面的例子中，仅保留第一条规则。

3.5 包含例外的规则

分类规则的一个自然扩展就是允许规则包含例外。它是在现有的规则上使用例外表达法来递增地修改一个规则集，而不需要重新建立整个规则集。例如，前面讨论过的鸢尾花问题，假如表3-1给出了一个新找到的花的数据，专家判断这个新的花是一个鸢尾花setosa的实例。如果用第1.2节给出的规则对花进行分类，那么下面两条规则将会得出错误结论:

```
If petal length ≥ 2.45 and petal length < 4.45 then Iris versicolor
If petal length ≥ 2.45 and petal length < 4.95 and
   petal width < 1.55 then Iris versicolor
```

将这两条规则进行修改，才能对新的实例进行正确分类。然而，只是简单地改变这些规则中的属性值的测试边界，并不能解决问题，因为用来建立规则集的实例也会被错分。对规则集的修改并不像听上去那么简单。

表3-1　一个新的鸢尾花

花萼长	花萼宽(cm)	花瓣长(cm)	花瓣宽(cm)	种类
5.1	3.5	2.6	0.2	?

首先专家需要给出解释，为什么新的花会和规则相抵触，根据得到的解释仅对相关的规则进行扩展，而不是修改现存规则中的测试。例如，上面两条规则中的第一条错将新的鸢尾花setosa 分到鸢尾花versicolor类里。可以利用其他一些属性建立一个例外，来取代修改规则中不等式里的边界值。

```
If petal length ≥ 2.45 and petal length < 4.45 then
   Iris versicolor EXCEPT if petal width < 1.0 then Iris setosa
```

这个规则表明如果花瓣长度在2.45~4.45cm之间，这种花就是鸢尾花versicolor，但是有个例外，如果同时花瓣的宽度小于1.0cm，那它就是鸢尾花setosa。

当然，可以在例外上使用例外，等等，将一个决策树的特征赋予规则集。除了可以用来对现存的规则集作递增的修改外，这些包含了例外的规则能够表达所有的概念描述。

图3-5所示的一组规则能够对鸢尾花数据集中的所有样本正确地分类（第1.2节）。这些规则一开始很难被理解，下面将一步一步地给予解释。首先选择一个缺省的输出类鸢尾花setosa，并显示在第一行。对这个数据集来说，缺省类的选择是任意的，因为每一种类型都有50个样本。通常是选择出现频率最高的类作为缺省类。

```
1  Default: Iris-setosa
2  except if petal-length ≥ 2.45 and petal-length < 5.355
3            and petal-width < 1.75
4         then Iris-versicolor
5              except if petal-length ≥ 4.95 and petal-width < 1.55
6                     then Iris-virginica
7                     else if sepal-length < 4.95 and sepal-width ≥ 2.45
8                          then Iris-virginica
9         else if petal-length ≥ 3.35
10             then Iris-virginica
11                  except if petal-length < 4.85 and sepal-length < 5.95
12                         then Iris-versicolor
```

图3-5 鸢尾花数据的规则

接下来的规则是在缺省的规则上给出例外。从第2行到第4行的第一个if...then给出了一个产生鸢尾花versicolor分类的条件。然而这个规则存在2个例外（从第5行到第8行），我们稍后处理。如果不符合第2行和第3行的条件，将转到第9行else，它表示了最初缺省类的第2种例外。如果符合第9行的条件，就属于类鸢尾花virginica（第10行）。从第11行到第12行是这一规则的另一个例外。

现在讨论第5行到第8行的例外情况。如果满足第5行或者第7行中的任何一个测试条件，那么鸢尾花versicolor的结论将被废除。这两个例外将得出相同的结论：鸢尾花virginica（第6行和第8行）。第11行和第12行是最后一个例外，当满足第11行的条件时，它废除了在第10行得到的鸢尾花virginica结论，最后产生的分类是鸢尾花versicolor。

在弄清楚这些规则如何阅读以前，需要花些时间仔细思考这些规则。尽管需要花一点时间，但在熟悉之后，解决except和if...then...else问题是轻而易举的。人们习惯于用规则、例外、和例外的例外来思考真实的问题，所以这也是表达一个复杂规则集的好方法。但是这种表达方法的最主要的优点是整个规则集的增长幅度适中。尽管对整个规则集的理解有点困难，但

是每一个单独的结论，每一个单独的then语句，只需要在那些导致它的规则和例外的范围里考虑。至于决策列，则需要重新审视前面所有的规则，来判断一个单独规则的确切影响。当开始理解大的规则集时，这种局域的特性是重要的。从心理上看，人们习惯把一个特定事件集或一种事件看成数据，当观察任何一个在例外结构里的结论时，以及当其中的一个事件转变成结论的一个例外时，增加一个except语句是解决问题的一个简单方法。

70~72 这里需要指出default...except if... then...结构，逻辑上与if...then...else...相等，else是无条件的，并且精确地指出缺省值是什么。当然一个无条件的else就是一个缺省值。（注意：上面的规则中没有无条件的else）。从逻辑上说，基于例外的规则可以简单地用if...then...else语句改写。采用例外形式来陈述，所获的益处更趋向于心理上的而不是逻辑上的。这里假设缺省值和较早出现的测试的应用范围，相对于以后的例外情况的应用范围更为广泛。如果真实情况确实如此，用户能够看到这是一个似乎可行的方法，用（普遍的）规则和（极少的）例外情况的表达方式比一个不同的、但是逻辑相同的结构，更容易被领会。

3.6 包含关系的规则

前面已经隐含地假设了规则中的条件涉及一个属性值和一个常量的测试。这类规则称为命题，因为使用属性-值语言定义的规则和逻辑学家所用的命题演算有相同的功能。在许多分类工作中，命题规则能够充分表达精练、正确的概念描述。例如前面所提到的天气、隐形眼镜的推荐、鸢尾花的类型以及劳工合同的接受程度的数据集，都能够用命题规则很好地表示。但是，在一些情况下一个更有表现力的规则形式将提供一个更加直观、简练的概念描述，这些情况包括了样本之间的关系，正如第2.2节遇见过的。

举一个具有代表性的例子，假设图3-6显示了一组8个不同形状和尺寸的积木，希望学到站立的概念。这是一个经典的二类问题，这两个类分别是站立(standing) 类和卧倒(lying) 类。其中4个有阴影的积木是肯定的概念样本——站立类，没有阴影的积木是否定的概念样本——卧倒类。学习算法的输入信息是积木的宽度、高度和每个积木的边数。表3-2显示了训练数据集。

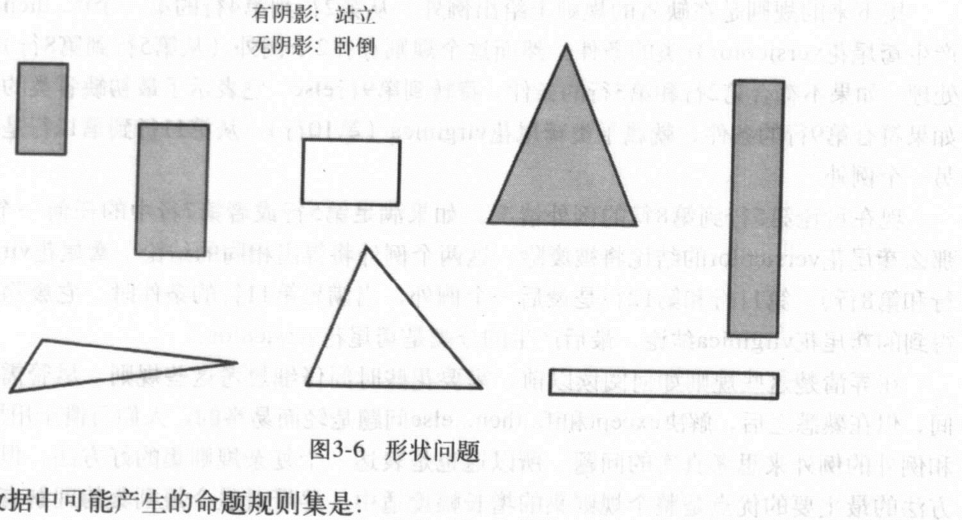


图3-6 形状问题

73 从这个数据中可能产生的命题规则集是：

```
if width ≥ 3.5 and height < 7.0 then lying
if height ≥ 3.5 then standing
```

表3-2 形状问题的训练数据

宽 度	高 度	边 数	分 类
2	4	4	站立
3	6	4	站立
4	3	4	卧倒
7	8	3	站立
7	6	3	卧倒
2	9	4	站立
9	1	4	卧倒
10	2	3	卧倒

为什么宽度的分界点是3.5？因为它是卧倒积木的最小宽度4，和高度小于7的站立积木的最大宽度3的平均值。同样7.0是高度的分界点，因为它是卧倒积木最大高度6，和宽度大于3.5的站立积木的最小高度8的平均值。将数值型的阈值设定为概念边界值的中间值是一个通用的方法。

尽管这两条规则能够在给出的样本上很好地运用，但是它们不是最好的方案。因为它们不能对许多新的积木进行分类（例如：积木的宽度是1，高度是2），还可以很容易地找出许多合理的而这两条规则却不适用的积木。

当人们在对这8个积木进行分类时，也许会发现“站立积木的高度大于宽度”。这条规则是在属性值之间进行比较，而不是将属性值与一个常量进行比较。

```
if width > height then lying
if height > width then standing
```

高度和宽度的真实值并不重要，重要的是它们之间的比较结果。这种形式的规则称为关系规则，因为它表示了属性之间的关系，而不是只针对一个属性实际情况的命题规则。 74

标准的关系包括名词性属性的等于或不等于运算，和数值属性的小于和大于的运算。尽管决策树里可以使用关系节点，就像能够在规则里使用关系条件一样，但是采用关系的方案里通常使用规则而不是树的表达形式。然而，大多数机器学习方案并不考虑关系规则，因为建立关系规则的代价太高。但是可以使用命题规则的方法来使用关系，就是新增一个属性即第二属性，来表示两个原始属性之间相等或不等的关系，如果是数值属性可以给出它们之间的差值。例如，可以在表3-2中增加一个是否宽度小于高度（width<height）的一个二元值属性。这些属性通常作为数据处理工作的一部分被加入。

经过看似较小的改进后，关系的知识表达能力能够得到极大的扩展。其中的奥秘是采用能使实例作用明确的方法来表示规则。

```
if width(block) > height(block) then lying(block)
if height(block) > width(block) then standing(block)
```

尽管这个例子似乎并没有得到很多扩展，但是如果能够把实例分解成多个部分，规则的表现能力确实能够得到扩展。例如，如果一个由一堆石块堆出的塔，一块堆在另一块上面，那么位于塔最顶端的石块是站立的，就可以用以下的规则表示：

```
if height(tower.top) > width(tower.top) then standing(tower.top)
```

这里tower. top是指最顶端的那块石块。到现在为止并没有获得任何益处。但是如果用tower. rest表示塔的其余部分，那么可以用下面的规则表示塔是由全部站立的石块组成。

```
if height(tower.top) > width(tower.top) and standing(tower.rest)
   then standing(tower)
```

看似很小的附加条件standing(tower. rest) 却是一个递归的表达形式，只有当塔的余下部分全部由站立的石块组成，附加条件standing(tower.rest)才能被满足。相同规则的递归应用将对此进行测试。当然，有必要增加一个如下的规则为递归设置一个适合的“最低点”。

```
if tower = empty then standing(tower.top)
```

使用这个附加条件，关系规则就能够表示那些不可能由命题形式表达的概念，因为递归能够应用于任意长的对象列上。像这样的规则集称为逻辑程序（logic programs），在机器学习领域里称为归纳逻辑编程（inductive logic programming）。这本书将不深入涉及这一
75 内容。

3.7 数值预测树

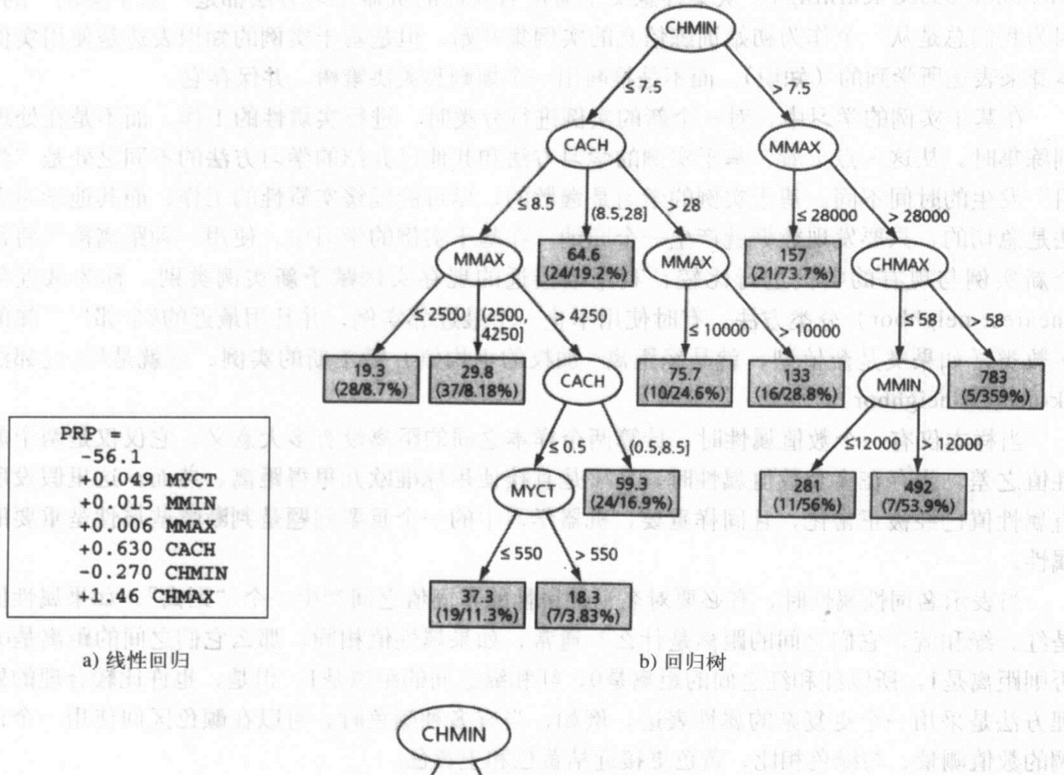
到现在为止所讨论的决策树和规则是设计用来预测实例的类而不是数量值。对像表1-5的CPU性能数据进行数量值的预测时，可以使用相同的决策树或者规则的表达形式，但是决策树的叶节点或者是规则的右边将包含一个数量值，这个数量值是叶节点或规则应用所涉及的全部训练集（类）值的平均值。统计学家使用回归（regression）这个术语表示计算一个能预测数值量的表达式的过程，因此在叶节点拥有平均数值的决策树称为回归树（regression tree）。

图3-7a是一个CPU性能数据的回归公式，图3-7b是一个回归树。在回归树的叶子上的数字是到达这个叶子的所有实例的平均类值。这个树比回归公式更大也更复杂，如果计算CPU性能预测值和实际测量值之间的平均绝对误差值时，将会发现由回归树预测得出的平均误差明显小于由回归公式计算的。回归树能做出更加精确的预测，是因为在这个问题上，一个简单线性模型的数据表达能力较差。然而，回归树的规模较大、很繁琐，也很难对它进行解释。

将回归公式和回归树相结合是一个可行的方案。图3-7c是一个在叶节点包含了线性公式，亦即回归公式，而不是一个预测值。这个树（稍有些困惑）称为模型树（model tree）。图3-7c包含的6个线性模型分别属于6个叶子，用LM1~LM6进行标记。模型树用多个线性“修补”（函数）来逼近连续函数，这是一种比线性回归或者回归树更好的表达形式。尽管模型树比回归树更小更容易理解，在训练数据上产生的平均误差值却要低。（然而，在第5章将看到在训练集上计算平均误差并不是一个评估模型性能的好方法。）

3.8 基于实例的表达

最简单的学习形式是简单地记住或者是死记硬背。一旦记住了一个训练实例集，在遇到一个新的实例时，就会在记忆中找出与之最相似的一个训练实例。唯一的问题是如何理解
76 “相似”，我们将很快对此进行解释。首先，注意这是采用一种完全不同的方法来表达从实例



a) 线性回归

b) 回归树

c) 模型树

图3-7 CPU性能数据模型 77

集里提取出的“知识”：保存实例本身，并且将类未知的新实例与现有的类已知的实例联系起来进行操作。这种方法直接在样本上工作，而不是建立规则。这就是基于实例的学习

(instance-based learning)。从某种意义上看所有其他的机器学习方法都是“基于实例”的，因为我们总是从一个作为初始训练信息的实例集开始。但是基于实例的知识表达是使用实例本身来表达所学到的（知识），而不是推断出一个规则集或决策树，并保存它。

在基于实例的学习中，对一个新的实例进行分类时，进行实质性的工作，而不是在处理训练集时。从这一点上看，基于实例的学习方法和其他已介绍的学习方法的不同之处是“学习”发生的时间不同。基于实例的学习是懒散的，尽可能延缓实质性的工作，而其他学习方法是急切的，只要发现数据就产生一个归纳。在基于实例的学习中，使用一种距离衡量将每个新实例与现有的实例进行比较，利用最接近的现存实例赋予新实例类别。称为最近邻(nearest-neighbor）分类方法。有时使用不止一个最近邻实例，并且用最近的k个邻居所属的多数类（如果类是数值型，就是经距离-加权的平均值）赋予新的实例。这就是k最近邻法(k-nearest-neighbor)。

当样本仅有一个数值属性时，计算两个样本之间的距离没有多大意义，它仅仅是两个属性值之差。当存在多个数值属性时，几乎是直接使用标准欧几里得距离。然而，这里假设所有属性值已经被正常化，且同样重要，机器学习中的一个重要问题是判断哪些属性是重要的属性。

当表示名词性属性时，有必要对名词性属性的不同值之间产生一个“距离”。如果属性值是红，绿和蓝，它们之间的距离是什么？通常，如果属性值相同，那么它们之间的距离是0，否则距离是1。所以红和红之间的距离是0，红和绿之间的距离是1。但是，也许比较合理的处理方法是采用一个更复杂的属性表达。例如，当有多种颜色时，可以在颜色区间使用一个色调的数值测量，与绿色相比，黄色更接近桔黄色和土黄色。

一些属性也许比另一些属性更加重要，这通常通过某种属性加权反映在距离度量上。从训练集上获得合适的属性权值，是基于实例学习中的一个关键问题。

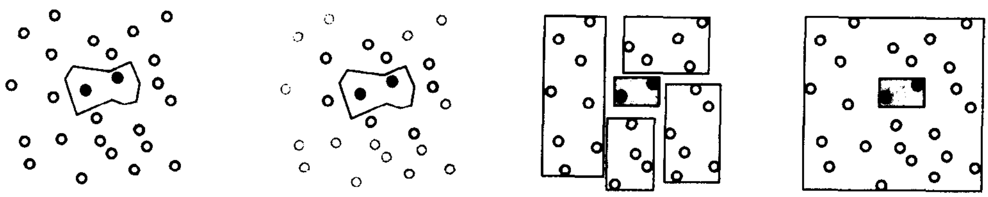
也许没有必要存储所有的训练实例。一方面是因为它可能使最近邻的计算过程异常缓慢，
78 另一方面它将不切实际地占用大量的存储空间。通常与类相对应的属性空间的部分区域比其他区域更稳定，所以在这些稳定的区域内只需要少数几个样本。例如，你也许可以期望类边界以内所需的样本密度要小于靠近类边界所需的密度。决定应该保留哪些实例，哪些实例应该抛弃是基于实例学习的另一个关键问题。

基于事例学习表达方式的弱点是，它不能对所学到的给出一个清晰的数据结构。从这方面说它和在这本书一开始所陈述的“学习”相冲突，实例并没有真正"描述"数据中的模式。然而，实例结合距离度量在实例空间刻划出的边界能够区别不同的类别，这是一种显式的知识表达形式。例如，给出两个属于不同类的实例，最近邻规则能有效地利用实例之间连线的垂直平分线将实例空间分裂开来。如果每个类都有几个实例，那么实例空间将被一组直线分隔开来，这组直线便是经过挑选的、属于一个类别的实例与另一个类别实例之间连线的垂直平分线。图3-8a里用一个9边形将属于实心圆的类从属于空心圆的类里分离出来。这个多边形隐含着最近邻规则的操作。

当训练实例被丢弃后，结果是每个类只保存几个有代表性的样本。图3-8b用深色空心圆圈显示的仅是几个真正在最近邻决策中使用到的样本，其他的样本（淡灰色的空心圆圈）可以被丢弃而不对结果产生任何影响。这些有代表性的样本就是一种显式的知识表达形式。

一些基于实例的表达法能够更进一步对实例进行明确的推广。典型的方法是通过建立矩

形区域来包围属于同一类的实例。图3-8c展示了可能产生的矩形区域。如果一个未知类的实例落入某一矩形区域之内，它将被赋予相应的类，而落在所有矩形区域以外的样本将服从最近邻规则。当然这将产生与直接的最近邻规则不同的决策边界，若将图3-8a 的多边形与矩形重叠后就会发现（此不同）。落入矩形的多边形部分将被砍掉，而由矩形边界取而代之。



a) b) c) d)

图3-8 分隔实例空间的不同方法 79

在实例空间的矩形推广法就像是包含特殊条件形式的规则，它对一个数值变量进行上、下边界的测试，并选择位于其间的区域。不同尺寸的矩形对应于由逻辑与组合在一起的在不同属性上的测试。选择一个最适合的矩形作为测试边界所产生的规则，将比由基于规则的机器学习方案产生的规则更为保守，因为对于区域的每一个边界，都有一个真正的实例落在边界上（或边界内）。而像$x < a$（x是一个属性值，a是一个常量）的测试将包围一半的空间，不管x有多小只要它小于a。当在实例空间运用矩形推广时，能够做到保守，因为如果一个新的样本落在所有区域以外，还可以求助于最近邻的度量方法。而采用基于规则的方法时，如果没有规则适用于这个样本，它将不能被分类，或是仅得到一个缺省的分类。更加保守的规则的优点是尽管保守的规则并不完整，但是它也许比一个覆盖所有事件的规则集的表达更为清楚。最后，要保证区域之间不能重叠，也就是保证最多只能有一个规则适合应用于一个样本，这样避免了在其他基于规则学习系统中，多个规则适用于一个样本的难题。

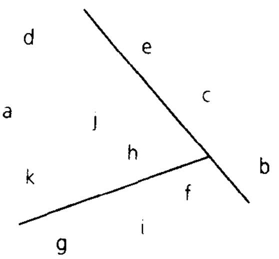
一个更复杂的推广是允许矩形区域嵌入在其他矩形区域中。正如图3-8d所示，基本上属于一个类的样本区域里包含了属于另一个不同类的内部区域。还可以允许嵌套内的嵌套，那么内部区域本身便可以包含一个不同类的内部区域，这个类也可能与最外面的区域属于同一个类。这种处理方法与第3.5节允许规则中有例外，以及例外的例外相类似。

这里需要指出在样本空间里，用边界的方法将基于实例学习可视化的技术有一点缺陷：它做了一个隐含的假设，假设属性是数值型的而不是名词性的。如果一个名词性属性的不同属性值被放置在一条直线上，那么在这条直线进行分段的推广是没有意义的：每个测试包含了一个属性值或者所有属性值（也许是属性值的任意一个子集）。尽管你能很容易或不太容易地将图3-8的样本想像成扩展到多维空间上，但是要想像包含了名词性属性的规则在多维实例空间上将是如何的，便困难多了。在许多场合里机器学习需要处理大量的属性，当扩展到高维实例空间时，直觉往往会导致我们步入歧途。 80

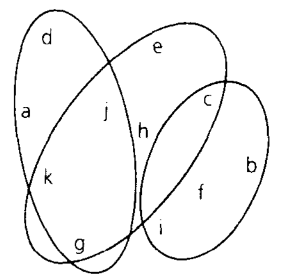
3.9 聚类

当机器学习学到的是聚类而不是一个分类器时，输出则采用一个显示实例如何落入聚类的图形形式。最简单的方法是让每个实例伴随一个聚类的编号，通过将实例分布在二维空间并且对空间加以分隔的形式来表示各个聚类，如图3-9a所示。

一些聚类的算法允许一个实例可以属于不止一个聚类，如Venn图，将实例分布在二维图形上，然后画出重叠的子集来表示每个聚类。另一些算法将实例与各个聚类的概率相关联而不是（直接）与类别相关联。从这个意义上说，每个实例存在一个对于各个聚类的成员归属的可能性或者程度，如图3-9c中所示。这个特殊的关联意味着一个概率问题，所以对于每个实例，所有概率和为1，尽管并不总是这样。其他算法产生一个分级的聚类结构，在结构顶层的实例空间被分为几个聚类，每个聚类将在下一层又被分为几个子聚类，如此下去。这样就产生如图3-9d所示的结构，聚类的成员在低层聚集的紧密程度要高于在高层聚集的程度。这种图称为系统树图（dendrogram）。这个术语与树图（tree diagram）有着相同的含义（希腊语dendron是"一棵树"），但是在聚类中似乎更倾向于使用古色古香的文言，也许是因为聚类技术首先运用的领域是生物物种，而在生物学领域通常使用古代语言对生物物种进行命名。



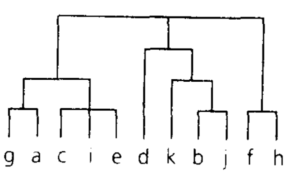
a)



b)

	1	2	3
a	0.4	0.1	0.5
b	0.1	0.8	0.1
c	0.3	0.3	0.4
d	0.1	0.1	0.8
e	0.4	0.2	0.4
f	0.1	0.4	0.5
g	0.7	0.2	0.1
h	0.5	0.4	0.1

c)



d)

81

图3-9　表示聚类的不同方法

聚类之后通常伴随着推导出一个决策树或规则集的步骤，从而将每个实例分配到它所属的聚类。这样说来，聚类操作只不过是通向结构描述的一个步骤。

3.10 补充读物

知识表达传统上是人工智能的一个重要主题，并在Brachman和Levesque（1985年）的一系列系统的论文中已得到了很好的论述。然而，这些论文都是关于如何表达手工制作出而非学习得来的知识，比较而言，可以从样本中学得的知识的表达方式（的论述）还相当原始。值得一提的是，在逻辑上被称为命题验算的命题规则的缺点，以及关系规则，或者称谓词验算（predicate calculus）的极大表现能力，在Genesereth和Nilsson（1987年）的第2章介绍逻辑部分得到很好的阐述。

我们提到了处理不同规则之间冲突的问题，对此有多种处理方案，称为冲突解决策略

(conflict resolution strategy)，已经开发运用于基于规则的程序系统。并在有关基于规则编程的书中阐述了有关内容，如Brownstown等（1985年）所著。然而，它们是为用于手工制作的而非学习来的规则集而设计的。手工制作的包含例外的规则在一个大型数据集上的应用，已经由Gaines和Compton（1995年）作了研究，并且Richards和Compton（1998年）探讨了它们作为经典的知识工程的替代的作用。

更多的有关概念表达的不同方法，可以在那些有关从样本中推出概念的机器学习方法的论文中找到。在第4章的补充读物和第6章的讨论部分将谈到这部分内容。 82

第4章　算法：基本方法

我们已经学习了如何表达输入和输出，现在来看机器学习算法。本章将介绍在实际数据挖掘中使用的、数据挖掘技术的一些基本概念。这里将不深入地考查一些很微妙的问题，比如高级的算法版本、可能的优化方法、在实际数据挖掘中产生出的复杂问题。这部分内容会在第6章进行阐述，届时将结合真正用于机器学习的实现方案进行讨论，譬如与本书配套的数据挖掘开发工具里的一些方案，以及在真实世界中的应用。深入理解这些更为复杂的问题是非常重要的，这样才能在分析某个具体数据集时，了解数据中真正发生了什么。

本章将研究一些基本概念。其中最有指导意义的一句话就是简单的方法通常能很好地工作。在分析实际数据集时，建议采用“简单优先”的方法论。数据集能够展示很多不同的、简单的数据结构形式。在一个数据集里，也许只有一个属性承担了所有的工作，而其他的都
83 是无关或冗余的属性。在另一个数据集里，所有属性也许是独立地、均等地对最终结果做出贡献。在第三个数据集里，也许拥有一个包含了几个属性的简单逻辑结构，这个结构可以由一个决策树得到。在第四个数据集里，也许存在一些独立的规则，能将实例划分到不同的类。在第五个数据集里，也许展示出不同的属性子集间的依赖性。在第六个数据集里，也许包含了一些数值属性间的线性依赖关系，关键是要为各个属性选择合适的权值，并求一个加权的属性值之和。在第七个数据集里，归类到实例空间的具体区域也许要受控于实例间的距离。在第八个数据集里，也许没有提供类值，学习是无指导的学习。

在具有无穷变化的数据集里，会产生很多不同的数据结构形式。要寻找某一种结构的数据挖掘工具，不管多有效，都可能会完全丢失其他不同结构的规律性，而这些结构是非常基本的。结果得到的是结构复杂的、难以理解的一种分类结构，而不是简单的、优美的、能够立刻被理解的另一种结构形式。

上面所描述的八种不同数据集形式中的每一个，都对应着一个适合于揭示它的不同的机器学习方案。本章将分别对这些结构进行讨论。

4.1　推断基本规则

这里有一个能从实例集里方便地找出非常简单的分类规则的方法，称为“1规则”（1-rule），简称1R。它产生一层的决策树，用一个规则集的形式表示，只在某个特定的属性上进行测试。1R是一个简单、廉价的方法，但常常能得到非常好的规则用以描述存在于数据中的结构。由它得出的简单规则经常能达到高得令人吃惊的正确率。也许这是因为真实世界的数据集中的数据结构相当基本，仅用一个属性就足以准确地判断出一个实例的类别。所以在任何事例上，首先采用最简单的方法总是一个好计划。

方法是：建立一个只对单个属性进行测试的规则，并进行不同的分支。每一个分支对应一个不同的属性值。分支的类就是训练数据在这个分支上出现最多的类。这种方法能够容易地计算出规则的误差率。只要计算在训练数据上产生的错误，即，统计不属于多数类的实例数量。

每一个属性都会产生一个不同的规则集，每条规则对应这个属性的每个值。对每一个属性的规则集的误差率进行评估，从中选出性能最好的一个。就是这么简单！图4-1是用伪代码形式表示的算法。 84

```
对于每个属性
    对于这个属性的每个属性值，建立如下的一条规则：
        计算每个类别出现的频率
        找出出现最频繁的类别
        建立规则，将这个类别赋予这个属性值
    计算规则的误差率
选择误差率最小的规则
```

图4-1 1R伪代码

表4-1 评估天气数据中的属性

	属 性	规 则	误 差	总误差
1	outlook	sunny → no	2/5	4/14
		overcast → yes	0/4	
		rainy → yes	2/5	
2	temperature	hot → no*	2/4	5/14
		mild → yes	2/6	
		cool → yes	1/4	
3	humidity	high → no	3/7	4/14
		normal → yes	1/7	
4	windy	false → yes	2/8	5/14
		true → no*	3/6	

*在出现两个相等结论时的一个任意选择

这里用表1-2的天气数据来研究1R是如何工作的（在讨论学习算法如何工作时，我们将多次采用这个数据）。为了在最后一列得到分类结果是否玩（play），1R将考虑四个规则集，一个属性对应一个规则集。表4-1列出了这些规则。星号表示采用了一个任意的选择，因为规则产生出两个可能性相等的结论。给出每个规则产生的错误分类的数量，以及整个规则集产生的错误分类的数量。1R选择所产生的规则集的错误数量最小的属性，就是第一和第三个规则集。可以从中任意选择一个规则集来打破这个平局而给出：

```
outlook: sunny    → no
         overcast → yes
         rainy    → yes
```

85

注意在一开始就没有对天气数据所涉及的活动做出特别说明。从得出的结论看非常奇怪，似乎在多云或者雨天才能进行这项活动，晴天却不适合。也许这是一项室内活动。

4.1.1 残缺值和数值属性

尽管1R是一个非常基本的学习方法，但是它可适用于残缺值和数值属性。1R处理残缺值和数值属性的方法既简单又高效。把残缺作为另一个属性值，例如，如果天气数据在阴晴属性上存在残缺值，那么在阴晴属性上产生的规则集将指定4个可能的类值，分别为sunny、overcast、rainy、第4个为残缺。

我们可以采用一个简单的离散方法将数值属性转换成名词性属性。首先，将训练样本按照数值属性的值进行排序、产生一个类值的序列。例如，根据温度属性值对数值版本的天气数据（表1-3）进行排序后产生的序列如下：

64	65	68	69	70	71	72	72	75	75	80	81	83	85
yes	no	yes	yes	yes	no	no	yes	yes	yes	no	yes	yes	no

离散通过在这个序列上放置断点来达到分隔。一个可行的方法是在类值发生变化之处放置断点，产生出8个范畴：

```
yes | no | yes yes yes | no no | yes yes yes | no | yes yes | no
```

将断点设置在两边样本之间的中间位置，即，64.5，66.5，70.5，72，77.5，80.5，84。然而，两个属性值为72的实例产生了一个问题，因为拥有相同温度属性值，却属于不同的类别。最简单的解决办法是将处于72的断点向右移一个，新的断点将是73.5，从而产生出一个混合的部分，其中no是多数类。

离散存在的一个严重问题是，有可能形成大量的类别范畴。1R算法将自然地倾向于选择能被分裂成很多范畴的属性，因为它会将数据集分裂成很多部分，所以实例与它们各自所在部分的多数类同属一类的可能性增大。事实上，一个极端的例子是每个实例中一个属性拥有一个不同的值。如标识码（identification code）属性表示实例是唯一的，它在训练数据上产生的误差率是0，因为每个部分只有一个实例。当然，高度分支的属性通常不能在测试样本上有
86 很好的表现，实际上标识码属性将不可能在训练实例以外的样本上产生正确地预测。这种现象被称为过度拟合（overfitting）。在第1章已经讨论过要避免过度拟合偏差（第1.6节），在以下的章节里我们将不断地遇到这个问题。

对于1R算法，当一个属性存在大量可能值时，过度拟合就很有可能发生。所以，当离散一个数值属性的时候，需要采用一条规则，这条规则规定了每个范畴上的多数类样本所须达到的最小数量。如果设置的最小样本数量是3，那么上面的分段将只剩下2个。分段过程将由以下形式开始：

```
yes no yes yes | yes . . .
```

在第一段里，确保多数类yes出现3次。然而紧接着的实例也是yes，所以将它包括进第一段也不会产生任何损失。新产生的分离结果是：

```
yes no yes yes yes | no no yes yes yes | no yes yes no
```

这样除了最后一段，每一段至少包括3个属于多数类的实例，通常在最后一段会出现少于3个多数类实例的情况。分隔的边界一般要落在两个不同类的样本之间。

当相邻的段拥有相同的多数类时，像第一和第二段，将它们合并之后并不会影响规则集的意义。所以，最终的离散结果是：

```
yes no yes yes yes no no yes yes yes | no yes yes no
```

从中产生的规则集是

```
temperature: ≤ 77.5 → yes
             > 77.5 → no
```

第二个规则包含一个任意的决定，这里选择no。如果选择yes，正如此例所示，将没有必要使用任何断点，使用相邻的类别来打破平局更加合适。实际上，这个规则在训练数据集上产生了5个错误，不如前面在阴晴属性上产生的规则有效。用同样的方法在湿度属性上产生的

规则如下：

```
humidity: ≤ 82.5 → yes
          > 82.5 and ≤ 95.5 → no
          > 95.5 → yes
```

这个规则在训练集上只产生了3个错误，它是在表1-3数据上最好的“1规则”。

最后，如果一个数值属性存在残缺值，便为它们建立一个额外的范畴，并且只在那些已经定义了属性值的实例上运用离散过程。 87

4.1.2 讨论

在一个标题为“简单的分类规则在大多数常用的数据集上表现良好”（Holte，1993年）的研讨会的论文中，报告了在16个数据集上对1R性能的深入研究，这16个数据集经常被机器学习研究人员用来评估他们的算法。在这项研究中使用了交叉验证法（cross-validation），这种评估技术将在第5章介绍，用来确保测试结果可作为从独立测试集上所能获得的结论的代表。经测试，在每一个数值属性段里，最小的样本数量被设为6，而不是上面所演示的3。

令人惊奇的是，尽管1R非常简单，但是它的表现却异常突出，甚至可以与经典的机器学习算法相媲美。在几乎所有的数据集上，它产生的正确率只比由经典的决策树归纳方案产生的决策树的正确率低几个百分点。这些决策树通常要比1R产生的规则要大很多。只对单个属性进行测试的规则往往可以成为一个复杂结构的替代，在确定了性能基线的情况下，建议采用“简单优先”的方法论，首先使用简单、基本的技术，然后再将它发展成更加精细的学习方案。显然，对精细学习方案所产生的结果进行解释较为困难。

1R方法学到了一个一层的决策树，它的叶子代表不同的类。一个表达力稍强的技术是对每类使用一个不同的规则。每一个规则是几个测试的逻辑与，每个测试与一个属性相对应。对于数值属性，测试将检查属性值是否在一个给定的区间里；对于名词性属性将检查属性值是否在一个属性值的子集里。有两种形式的测试，区间和子集，是从属于每个类的训练数据上学到的。对于数值属性，区间的端点是那个类别的训练数据上出现的最大和最小值。对于名词性属性，子集只包含与类相对应的实例里该属性出现的值。表达不同类的规则经常会重叠，在预测阶段，使用满足测试条件最多的规则进行预测。这些简单的技术通常能对一个数据集给出有用的第一印象。它的处理速度极快，能够在极其庞大的数据上应用。

4.2 统计建模

1R方法使用单个属性作为决策的依据，并且选择其中工作性能最好的那个属性。另一个简单技术是对于一个给定的类，使用所有属性，让它们对决策做出同等重要、彼此独立的贡献。当然，这是不现实的，现实的数据集里的属性并不同等重要，也不彼此独立。但是它引出一个简单方案，并且在实际中表现极佳。 88

表4-2是一个天气数据的汇总，它统计了每种属性值配对和玩的每个属性值（yes 和no）一同出现的次数。例如，从表1-2可以发现，阴晴属性为sunny的5个样本中，2个样本的玩＝yes，3个样本的玩＝ no。表4-2第一行为每个属性所有可能值简单记录了这样的出现次数，最后一列的玩数字统计了yes和no总共出现的次数。表的下半部分是以分数形式，或用观察到的概率改写了同样的信息。例如，玩是yes的天数是9，其中阴晴是sunny有2天，由此产生的分数是2/9。对于（表中的）玩列，分数则有不同的含义，它们分别为玩属性值是yes和no的天数

在总天数中所占的百分率。

表4-2 拥有统计数和概率的天气数据

	阴晴			温度			湿度			刮风		玩	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2	3	hot	2	2	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

表4-3 新的一天

阴晴	温度	湿度	刮风	玩
sunny	cool	high	true	?

假设遇到如表4-3所示的一个新样本。我们认为表4-2中的5个属性：阴晴，温度，湿度，刮风和玩为yes或no的总体似然（likelihood），是同等重要、彼此独立的，并将与其对应的分数相乘。察看结果为yes（的情形）将给出：

$$yes\text{的似然} = 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053$$

89 这些分数是根据新的一天的属性值，从表4-2中取出与yes相对应的值，最后的9/14是玩为yes的天数占总天数（14天）的百分率。对结果为no的相似计算将得到：

$$no\text{的似然} = 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0206$$

可以看出，对于这个新的一天，玩是no的可能性是yes的4倍。通过正常化将这两个结果转换成概率，使它们的概率之和为1。

$$yes\text{的概率} = \frac{0.0053}{0.0053 + 0.0206} = 20.5\%$$

$$no\text{的概率} = \frac{0.0206}{0.0053 + 0.0206} = 79.5\%$$

这个简单且直观的方法基于有条件概率的贝叶斯规则。贝叶斯规则指出，如果存在一个假说H，和基于假说的例证E，那么

$$\Pr[H \mid E] = \frac{\Pr[E \mid H]\Pr[H]}{\Pr[E]}$$

$\Pr[A]$指事件A发生的概率，$\Pr[A|B]$是基于另一事件B发生，事件A发生的概率。假说H为玩的结果是yes，那么$\Pr[H|E]$将是20.5%，正如前面计算所得。例证E是新的一天的属性值的特定组合：阴晴 = sunny、温度= cool、湿度= high、刮风= true。这4个例证分别用E_1、E_2、E_3和E_4表示。假设这些例证是独立的（对于给出的类），将概率相乘后就得到它们的组合概率：

$$\Pr[yes \mid E] = \frac{\Pr[E_1 \mid yes] \times \Pr[E_2 \mid yes] \times \Pr[E_3 \mid yes] \times \Pr[E_4 \mid yes] \times \Pr[yes]}{\Pr[E]}$$

不用担心分母部分，不需要考虑分母，因为分母会在最后的正常化步骤（使yes和no的概率之

和为1）里被消除。最后的Pr[*yes*]是在不知道任何例证*E*的情况下，结论是yes的概率，也就是对于所涉及的特定日期的情况一无所知，称为假说*H*的先验概率（prior probability）。在这个例子里，是9/14，因为14个训练样本里有9个样本的玩属性值是yes。将表4-2里的分数替换成适合的例证概率得到：

$$\Pr[yes \mid E] = \frac{2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14}{\Pr[E]}$$ 90

和前面计算的一样。分母Pr[*E*]将在做正常化时消失。

这种方法称为朴素贝叶斯（Naïve Bayes），因为它基于贝叶斯规则并"朴素"地假设（属性）独立。只有当事件彼此独立时，概率相乘才是有效的。属性独立的假设，在现实生活中肯定是过于简单的假设。尽管有些名不符实，但在实际数据集上进行测试时，朴素贝叶斯工作得非常好，特别是当与一些在第7章将要介绍的属性选择程序相结合后，属性选择程序可去除数据中的一些冗余造成非独立的属性。

如果某个属性值没有联合每一个类值一起出现在训练集里，那么朴素贝叶斯法将会出错。假设，在一个不同的天气数据中，所有训练数据的属性值阴晴=sunny，总是伴随着结论no。那么属性值阴晴=sunny是yes的概率Pr[*outlook=sunny* | yes]是0，因为其他的概率将与这个0相乘，所以不管其他的概率有多么大，最终yes的概率都将是0。概率0超过其他的概率掌握了否决权。这不是一个好现象。然而，可根据频率来计算概率的方法，进行一些小的调整，便可很容易地弥补这个缺陷。

例如，正如表4-2上半部分所示，对于玩=yes，有2个样本的阴晴是sunny；4个样本的阴晴是overcast，3个样本的阴晴是rainy，下半部分给出了这些事件的概率，分别是2/9、4/9和3/9。我们可以在每一个分子上加1，并且在分母上加3进行补偿，所以得到的概率分别为3/12，5/12和4/12。这将保证当一个属性值出现0次时，得到一个很小但是非0的概率。在每一个计数结果上加1的方法是一个标准的技术，称为拉普拉斯估计器（Laplace estimator），它出自18世纪伟大的法国数学家Pierre Laplace。尽管它在实际中能很好地工作，但是也没有特别的理由需要在计数结果上加1。取而代之，可以使用一个很小的常量μ：

$$\frac{2+\mu/3}{9+\mu},\ \frac{4+\mu/3}{9+\mu} \text{ 和 } \frac{3+\mu/3}{9+\mu}$$

这里将μ设为3，它有效地提供了一个权值，这个权值决定了一个先验值（1/3，1/3和1/3）对每个可能属性值的影响力。与训练数据集的一个新的例证相比，一个大的μ值说明这些先验值是非常重要的，而小的μ值则说明先验值的影响力较小。最后，在分子部分将μ平均地分成3份并没有特别的理由，所以可以使用以下形式替代：

$$\frac{2+\mu p_1}{9+\mu},\ \frac{4+\mu p_2}{9+\mu} \text{ 和 } \frac{3+\mu p_3}{9+\mu}$$ 91

这里p_1、p_2和p_3之和为1。这3个数值分别是属性阴晴的值为sunny、overcast和rainy时的先验概率。

现在它是一个完整的贝叶斯公式，先验概率已经分配到每一个所见到的属性值。它的优点是十分严密，但缺点是通常并不清楚应该如何分配先验概率。在实践中，只要训练实例的数量合理，使用不同的先验概率几乎没有差别。人们通常用拉普拉斯估计器估计频率，将计数结果初始化为1而不是0。

4.2.1 残缺值和数值属性

使用贝叶斯公式的一个优势是：处理残缺值并不是难题。例如，如果表4-3样本中阴晴的属性值是残缺值，计算时只需要省略这个属性，结果是：

$$yes\text{的似然}=3/9\times 3/9\times 3/9\times 9/14=0.0238$$

$$no\text{的似然}=1/5\times 4/5\times 3/5\times 5/14=0.0343$$

这两个值分别比前面的计算值要高出很多，原因是缺少了其中的一个分数。但这并不会成为问题，因为两个公式都缺少一个分数，这些似然还要被正常化。最终产生的yes和no的概率分别是41%和59%。

在一个训练实例中，如果一个属性值残缺，它便不被包括在频率计算中，概率的计算取决于真正出现属性值的训练实例的个数，而不是训练实例的总数。

在处理数量值时，通常把它们假设成拥有“正态”（normal）或者“高斯”（Gaussian）的概率分布形式。表4-4对天气数据做了总结，其中数值属性数据来源于表1-3。对于名词性属性的计算方法和以前一样，对于数值属性，只需列出所有出现的值。然后，名词性属性的统计值经正常化成为概率；而数值属性，则计算出每一个数值属性在每一个类上的平均值（mean）和标准差（standard deviation）。所以，所有类值为yes的实例在属性温度上的平均值是73、标准差是6.2。这里的平均值是（数值属性在同一个类上的）平均属性值，也就是（在同一个类上的）属性值之和除以属性值的个数。标准差是样本方差（variance）的平方根值，计算方法
92 为：将每一个属性值减去平均值后进行平方，然后求和，最后除以属性值的个数减1。在求出这个样本的方差后，再对方差取平方根得到标准差。这是对一个数据集计算平均值和标准差的标准方法（“减1”是为了得到样本的自由度，这是一个统计学概念，这里将不再深入介绍）。

表4-4 数值天气数据和统计结果

	阴晴			温度			湿度			刮风		玩	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2	3		83	85		86	85	false	6	2	9	5
overcast	4	0		70	80		96	90	true	3	3		
rainy	3	2		68	65		80	70					
				64	72		65	95					
				69	71		70	91					
				75			80						
				75			70						
				72			90						
				81			75						
sunny	2/9	3/5	平均	73	74.6	平均	79.1	86.2	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	标准差	6.2	7.9	标准差	10.2	9.7	true	3/9	3/5		
rainy	3/9	2/5											

对于一个平均值为μ和标准差为σ的正态分布，它的概率密度函数表达式看似有些令人生畏。

$$f(x)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

不必害怕！它意味着，当属性温度有一个值，譬如说66，如果考虑yes 的结果，只需将$x=66$，$\mu=73$和$\sigma=6.2$代入公式。所以由概率密度公式得出的值为：

$$f(temperature=66 \mid yes)=\frac{1}{\sqrt{2\pi}\cdot 6.2}e^{-\frac{(66-73)^2}{2\cdot 6.2^2}}=0.0340$$

当属性湿度的值是90时，使用相同方法计算结论为yes的概率密度：

$$f(humidity=90 \mid yes)=0.0221$$

某个事件的概率密度函数与它的概率是密切相关的。然而，它们并不是一回事。如果温度是连续的，那么当温度正好是66或是其他任何值，比如63.141 592 62时的概率就为 0 。概率密度函数$f(x)$的真正含义是指这个量落在x附近的一个很小区域里，譬如说在$x-\varepsilon/2$和$x+\varepsilon/2$之间的概率是$\varepsilon f(x)$。如果温度测量是（与真实值）最为接近的度数，湿度测量也是（与真实值）最为接近的百分点，那么上面所得到的便是正确的。当使用这些概率时，你也许会认为应该要乘以精确值ε，然而没有必要。因为同样的ε会出现在yes和no的似然中，并且在计算概率时被消除。 93

对表4-5所列的新的一天运用这些概率，得到：

$$yes的似然=2/9\times 0.0340\times 0.0221\times 3/9\times 9/14=0.000\,036$$

$$no的似然=3/5\times 0.0221\times 0.0381\times 3/5\times 5/14=0.000\,108$$

由此产生的概率为：

$$yes的概率=\frac{0.000\,036}{0.000\,036+0.000\,108}=25.0\%$$

$$no的概率=\frac{0.000\,108}{0.000\,036+0.000\,108}=75.0\%$$

这些数值与先前对表4-3所示的新的一天的概率计算结果非常接近，因为属性温度和湿度的属性值分别为66和90，与前面使用cool和high分别作为这两个属性值时计算出的概率相似。

表4-5 另一个新的一天

阴 晴	温 度	湿 度	刮 风	玩
sunny	66	90	true	?

正态分布的假设很容易将朴素贝叶斯分类器进行扩展，使它能够处理数值属性。如果任何数值属性的值有残缺，那么平均值和标准差的计算仅基于现有的属性值。

4.2.2 用于文档分类的贝叶斯模型

机器学习的一个重要领域是文档分类，其中每一个实例就是一个文档，而实例的类就是文档的主题。如果文档是新闻，那么它的类也许可以是国内新闻、海外新闻、财经新闻和体育新闻。文档的特性是由出现在文档中的单词所描述的，一种应用机器学习对文档分类的方法是用布尔值属性表示每个单词的出现或者空缺。在文档分类的应用方面朴素贝叶斯是一个深受欢迎的技术，因为它的处理速度快而且正确率高。

然而，朴素贝叶斯法忽略了每个单词在文档中出现的次数，而在决定一个文档的分类时，这些信息拥有潜在的重要价值。取而代之，一个文档可以被看成是一袋单词（a bag of words）—— 94
即一个集合，这个集合包含文件中所有单词，在文件中多次出现的单词在集合中也多次出现

（从技术上说，一个集合所包含的每个成员应是唯一的，而一袋子可以拥有重复的成员）。采用一个修改过的朴素贝叶斯便可利用单词的频率，这个修改过的朴素贝叶斯有时称为多项朴素贝叶斯。

假设n_1，n_2，…，n_k是单词i在文件中出现的次数，P_1，P_2，…，P_k是从所有H类文档中取样得到单词i的概率。假设概率与单词的上下文以及单词在文件中的位置无关。这些假设产生出一个文档概率的多项式分布（multinomial distribution）。在这种分布中，对于一个给定类别H，文件E的概率，换句话说，计算贝叶斯规则中$\Pr[E \mid H]$的公式是

$$\Pr[E \mid H] \approx N! \times \prod_{i=1}^{k} \frac{P_i^{n_i}}{n_i!}$$

这里，$N=n_1+n_2+\cdots+n_k$是文件中单词的数量。使用阶乘的原因是考虑到根据单词袋（bag-of-words）模型中每个单词出现的次序并不重要。P_i是通过计算在所有属于类别H的训练文档中，单词i出现的相对频率而得到的估计。实际上，应该再有一项，即类别H的模型产生出与E长度相等的文件的概率，（这就是为什么要用≈而不是＝），但是通常假设对于所有类别来说（这个概率）都是相等的，因此可以忽略这一项。

例如，假设单词表里只有两个单词：黄（yellow）和蓝（blue），一个特定文档类别H存在$\Pr[yellow \mid H]=75\%$，$\Pr[blue \mid H]=25\%$（你也许称类别H为黄绿（yellowish green）类文档）。假如E是文档blue yellow blue，长度$N=3$个单词。存在4个可能的、拥有3个词的单词袋。一个是{yellow yellow yellow}，根据上面得出的公式得到它的概率为：

$$\Pr[\{yellow\ yellow\ yellow\} \mid H] \approx 3! \times \frac{0.75^3}{3!} \times \frac{0.25^0}{0!} = \frac{27}{64}$$

其他3个的概率分别是：

$$\Pr[\{blue\ blue\ blue\} \mid H] = \frac{1}{64}$$

$$\Pr[\{yellow\ yellow\ blue\} \mid H] = \frac{27}{64}$$

95

$$\Pr[\{yellow\ blue\ blue\} \mid H] = \frac{9}{64}$$

这里的E与最后一种情况相对应（注意：在一袋单词中的单词次序可以忽略）；所以由黄绿文档模型产生出E的概率是9/64，或14%。假如另一个类，蓝绿（very bluish green）文档（称它为H'），拥有的概率$\Pr[yellow \mid H']=10\%$，$\Pr[blue \mid H']=90\%$。由这个模型产生出E的概率为24%。

如果只有这两个类，是不是意味着E是蓝绿文档类呢？并不一定。前面给出的贝叶斯规则说，必须考虑每一个假说的先验概率。如果你实际上知道蓝绿文档罕见程度是黄绿文档的两倍，这将足够胜过14%和24%的不同，倾向于黄绿类文档。

前面概率公式中的阶乘并不需要真正计算，因为它对于每一个类是一样的，所以会在正常化的过程中被消除。然而，在公式里仍然需要将很多小概率相乘，这将迅速产生出非常小的数值，从而在大的文挡上造成下溢。不过这种问题可以通过对概率取对数替代概率本身来避免。

在多项朴素贝叶斯公式里，判断一个文档的类不仅要根据文档中出现的单词，还要根据

这些单词在文档中出现的次数。对于文档分类来说，多项朴素贝叶斯模型的性能通常优于普通的朴素贝叶斯模型，尤其在大型字典级的文档上表现尤其突出。

4.2.3 讨论

朴素贝叶斯法给出了一个简单并且概念清晰的方法，来表达、使用和学习概率的知识。使用它能够达到很好的预测结果。在许多数据集上，朴素贝叶斯的性能能与一些更加成熟的分类器相媲美，甚至会有更出色的表现。有一句格言是，始终从简单的方法入手。同样地，在机器学习领域，人们不断地努力使用更精细的学习方案想获得好的预测结果，但是最终只是在几年后发现，那些简单的方法，如1R和朴素贝叶斯能够得到同样好、甚至更好的结果。

朴素贝叶斯法在很多数据集上的表现差强人意，其中的原因很容易发现。因为朴素贝叶斯处理属性的时候，认为属性之间是完全独立的，所以一些冗余的属性会破坏机器学习过程。一个极端的例子是，如果在天气数据中加入一个新的属性，该属性拥有与属性温度相同的值，那么属性温度的影响力将会增加：属性温度的所有概率将被平方，在最后的决策上具有更大的影响力。如果在数据集中加入10个这样的属性，那么最终的决策将仅根据属性温度而做出。属性 96
之间的依赖性不可避免地会降低朴素贝叶斯识别数据中究竟发生什么的能力。然而，这种情况可以通过在决策过程中，采用仔细挑选属性子集的方法来避免。第7章将阐述如何选择属性。

对于数值属性，正态分布的假设是朴素贝叶斯的另一个限制，我们在这里进行一些说明。许多属性值并不呈正态分布。然而，对于数值属性，我们也可以采用其他分布形式：正态分布并没有特殊的魔力。如果你知道一个特定的属性可能遵循其他的分布形式，可以使用那种分布形式的标准估计过程。如果你怀疑数值分布不是正态分布，又不知道真正的分布形式，可以使用“核密度估计”（kernel density estimation）过程，核密度估计并不把属性值的分布假设成任何特定形式的分布。另一种可行的处理方法是首先将数据离散。

4.3 分治法：创建决策树

创建决策树的问题可以用递归形式表示。首先，选择一个属性放置在根节点，为每一个可能的属性值产生一个分支。这将使样本集分裂成多个子集，一个子集对应于一个属性值。然后在每一个分支上递归地重复这个过程，仅使用真正到达这个分支的实例。如果在一个节点上的所有实例拥有相同的类别，即停止该部分树的扩展。

唯一存在的问题是，对于一个给定的、拥有不同类别的样本集，如何判断应该在哪个属性上进行分裂。再次来看天气数据，每次分裂都存在4种可能性，在最顶层产生的树如图4-2所示。哪个才是最好的选择呢？类分别是yes和no的实例数量显示在叶子上。当叶子只拥有单一类别yes或no时，将不必继续分裂，到达那个分支的递归过程也将停止。因为我们要寻找较小的树，所以希望递归过程尽早停止。如果能够测量每一个节点的纯度，就可以选择能产生最纯子节点的那个属性进行分裂。观察图4-2，然后思考哪个属性是最佳选择。

我们将要使用的纯度量度被称为信息量（information），量度单位是位（bit）。纯度量度与树的一个节点相关联，代表了一个期望信息总量，用于说明到达这个节点的新实例将被分到yes还是no类所需的信息总量。不同于计算机内存所用的位，这里所期望的信息量通常牵涉到1位中的部分——经常小于1！根据在那个节点上yes和no类的实例数量来计算。我们很快将在后面看到计算的具体细节。首先讨论如何使用它。当对图4-2的第1个树进行评估时，在叶 97

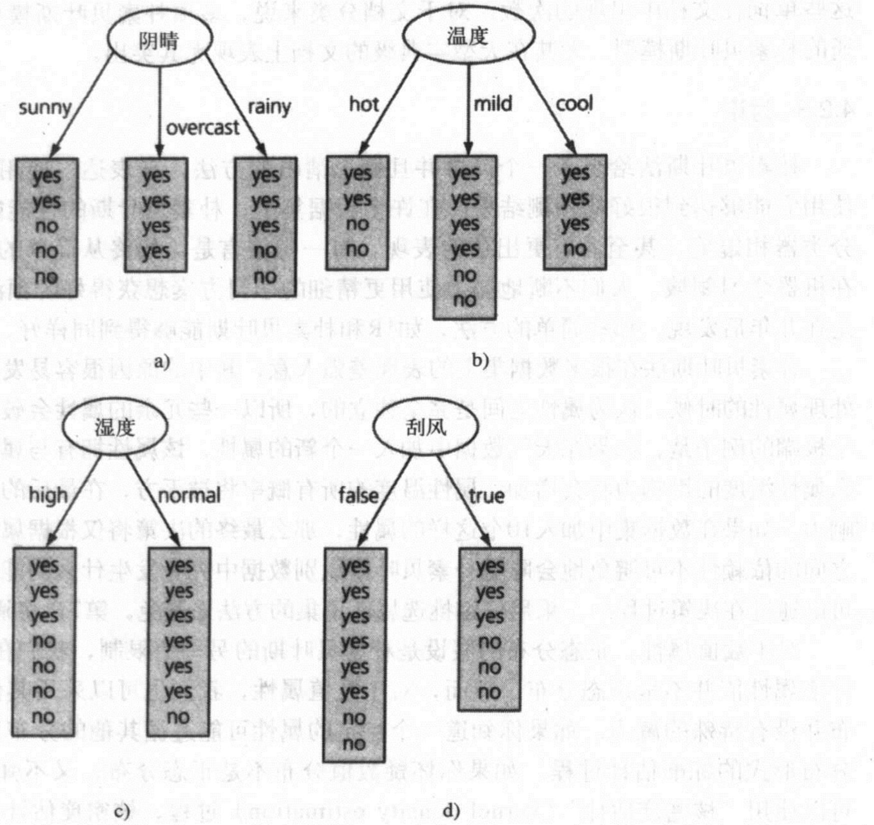


图4-2　天气数据树的树桩

节点上的yes和no类的实例数量分别是[2，3]、[4，0]和[3，2]，因此，这些节点上的信息值分别是：

info([2，3])=0.971 位

info([4，0])=0.0 位

info([3，2])=0.971 位

计算它们的平均信息值，并考虑到达每个分支的实例的数量：有5个实例到达第1和第3分支；4个实例到达第2分支。

info([2，3]，[4，0]，[3，2])=(5/14) × 0.971+(4/14) × 0+(5/14) × 0.971=0.693位

这个平均值代表了期望的信息总量，它是由指给定如图4-2a所示的树结构，对一个新实
98 例的类别进行说明所必需的信息量。

在任何初始树（图4-2所示）创建之前，处于根节点的训练样本由9个yes和5个no组成，与之相对应的信息值是

info([9，5])=0.940 位

因此图4-2a树所获的信息增益（information gain）为：

gain(outlook)=info([9，5]) − info（[2，3]，[4，0]，[3，2]）=0.940 − 0.693=0.247 位

它能够解释成在属性阴晴上建立一个分支的信息值。

随后的方法就很清楚了。为每一个属性计算信息增益，选择获得最多信息量的属性进行

分裂。图4-2的信息增益分别是：

gain(outlook)=0.247 位
gain(temperature)=0.029 位
gain(humidity)=0.152 位
gain(windy)=0.048 位

所以在根节点选择阴晴作为分裂属性。希望这个最佳选择与你的直觉相一致。它是唯一能获得一个全纯子节点的选择，这为阴晴属性超越其他所有属性赢得了相当大的优势。湿度属性是第2个最佳选择，因为它产生了一个几乎是全纯且较大的子节点。

接着继续进行递归过程。图4-3显示了在阴晴属性值为sunny的节点上的进一步分支的可能性。很明显，在属性阴晴上的再次分裂不会发生任何新的变化，所以仅考虑其他3个属性。在这3个属性上产生的信息增益分别为：

gain(temperature)=0.571 位
gain(humidity)=0.971 位
gain(windy)=0.020 位

因此选择湿度属性作为在这一节点的分裂属性。在随之产生的子节点上并不需要进一步分裂，所以这个分支就结束了。

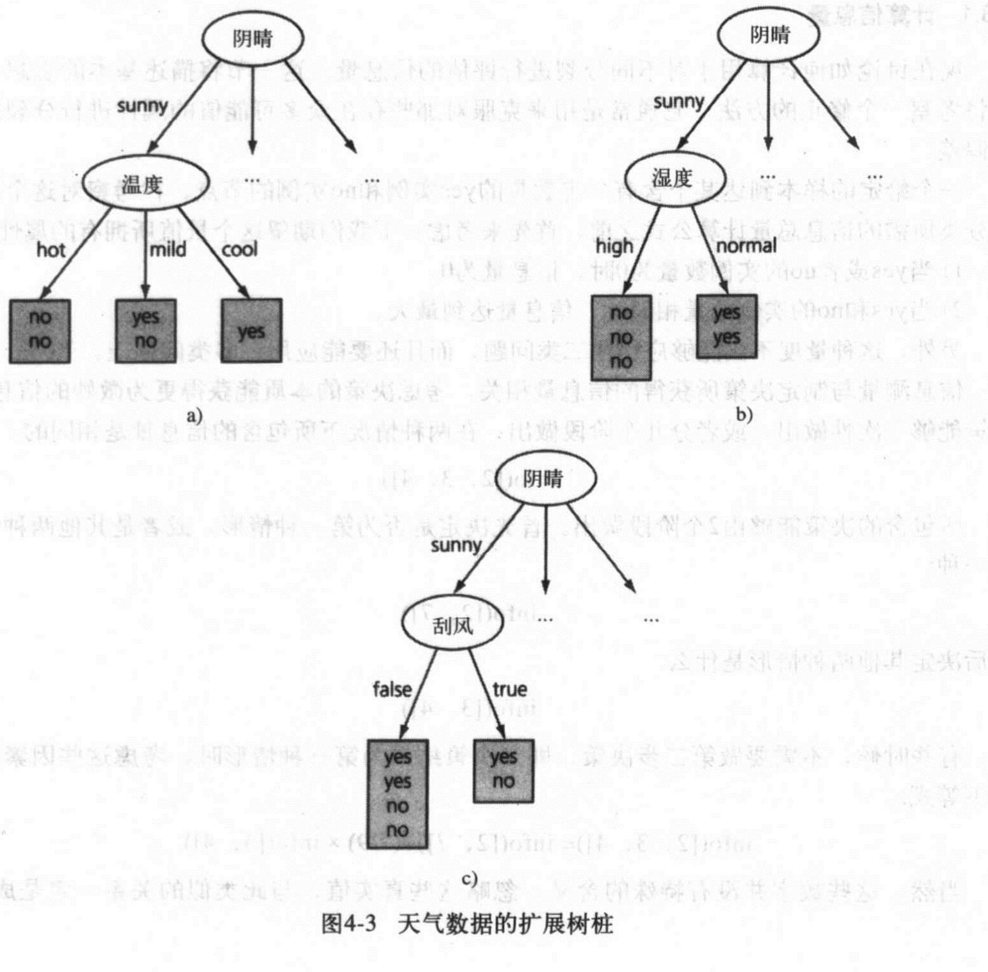


图4-3 天气数据的扩展树桩

继续应用这样的思想方法，将产生关于天气数据的决策树，如图4-4所示。理想的停止条件是所有叶节点都是纯的，也就是当叶节点包含的实例拥有相同的类别时。然而，也许并不可能达到这种理想状态，因为当训练集里包含2个拥有相同属性值，但是属于不同类别的样本时，递归过程将不可能停止。所以，停止条件应为当数据不能被进一步分裂时。

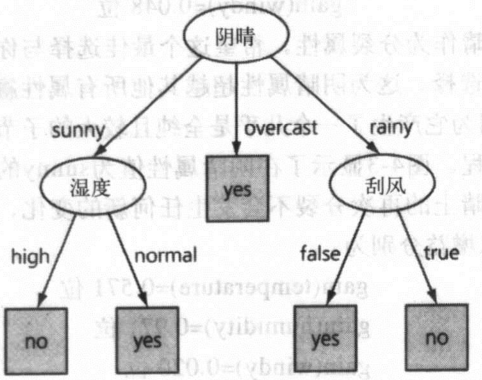


图4-4　天气数据的决策树

4.3.1　计算信息量

现在讨论如何计算用于对不同分裂进行评估的信息量。这一节将描述基本的思想，下一节将考察一个修正的方法，它通常是用来克服对那些存在众多可能值的属性进行分裂选择时的偏差。

一个给定的样本到达某个含有一定数量的yes实例和no实例的节点，在考察对这个样本进行分类所需的信息总量计算公式之前，首先来考虑一下我们期望这个量值所拥有的属性：

1) 当yes或者no的实例数量为0时，信息量为0。

2) 当yes和no的实例数量相同时，信息量达到最大。

另外，这种量度不仅能够应用在二类问题，而且还要能应用在多类问题上。

信息测量与制定决策所获得的信息量相关，考虑决策的本质能获得更为微妙的信息特性。决策能够一次性做出，或者分几个阶段做出，在两种情况下所包含的信息量是相同的。例如：

info([2，3，4])

所包含的决策能够由2个阶段做出。首先决定是否为第一种情形，或者是其他两种情形中的一种：

info([2，7])

然后决定其他两种情形是什么：

info([3，4])

有些时候，不需要做第二步决策，即当决策结果为第一种情形时。考虑这些因素后产生如下等式：

99~101　info([2，3，4])= info([2，7])+(7/9) × info([3，4])

当然，这些数字并没有特殊的含义，忽略这些真实值，与此类似的关系一定是成立的。

由此可以在先前的清单上增加一条标准:

3) 信息必须遵循如上所示的多阶段特性。

令人惊喜的是只用一个函数便能满足所有这些特性，称为信息值（information value）或者熵（entropy）:

$$\text{entropy}(p_1, p_2, \cdots, p_n) = -p_1 \log p_1 - p_2 \log p_2 \cdots - p_n \log p_n$$

使用负号是因为分数p_1, p_2, …, p_n 的对数值为负，因此熵实际是正数。一般对数的底数取2，熵是以“位”为单元的，就是通常在计算机上使用的位。

熵公式里的参数p_1, p_2, …, p_n为分数，它们的和为 1 ，例如

$$\text{info}([2,\ 3,\ 4]) = \text{entropy}(2/9,\ 3/9,\ 4/9)$$

多级的决策特性通常写成如下形式:

$$\text{entropy}(p, q, r) = \text{entropy}(p, q+r) + (q+r) \cdot \text{entropy}\left(\frac{q}{q+r}, \frac{r}{q+r}\right)$$

这里$p+q+r = 1$。

因为采用对数计算方法，信息量度计算不需要算出各个分数，例如

$$\text{info}([2,3,4]) = -2/9 \times \log 2/9 - 3/9 \times \log 3/9 - 4/9 \times \log 4/9$$
$$= [-2\log 2 - 3\log 3 - 4\log 4 + 9\log 9]/9$$

这是在实际中通用的信息量度计算方法。所以图4-2第一个树的第一个叶节点的信息值是:

$$\text{info}([2,3]) = -2/5 \times \log 2/5 - 3/5 \times \log 3/5 = 0.971\text{位}$$

与在先前（第4.3节）得到的结果是一样的。

4.3.2 高度分支属性

当一些属性拥有的可能值的数量很大，从而使分支的路径增加，产生出很多子节点时，计算信息增益就会出现一个问题。这个问题可以通过一个极端的例子来说明，当数据集的某个属性对于每一个实例存在一个不同属性值时，譬如，一个标识码属性。 102

表4-6 带有标识码的天气数据

标识码	阴 晴	温 度	湿 度	刮 风	玩
a	sunny	hot	high	false	no
b	sunny	hot	high	true	no
c	overcast	hot	high	false	yes
d	rainy	mild	high	false	yes
e	rainy	cool	normal	false	yes
f	rainy	cool	normal	true	no
g	overcast	cool	normal	true	yes
h	sunny	mild	high	false	no
i	sunny	cool	normal	false	yes
j	rainy	mild	normal	false	yes
k	sunny	mild	normal	true	yes
l	overcast	mild	high	true	yes
m	overcast	hot	normal	false	yes
n	rainy	mild	high	true	no

表4-6给出了带有额外属性的天气数据。图4-5对标识码属性进行分裂而产生树桩。给定这个属性的值，要说明它的类别所需的信息量是：

info([0，1])+info([0，1])+info([1，0])+ ⋯+info([1，0])+info([0，1])

由于14项中的每一项都是0，所以信息量为0。这个结果并不奇怪：标识码属性能够区别每个实例，所以它能确定类别，而不会出现任何模棱两可的情况，如表4-6所示。所以这个属性的信息增益就是在根节点上的信息量，即info([9，5])=0.940 位。它比在其他任何属性上获得的信息增益值要大，毫无疑问标识码将被选为分裂属性。但是在标识码属性上的分支对预测未知实例的类别并没有任何帮助，也没能描述任何有关决策的结构，而这两点正是机器学习的双

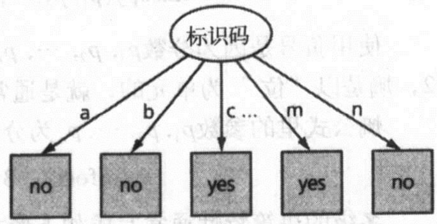


图4-5　标识码属性的树桩

103 重目标。

由此可见，采用度量信息增益的方法会倾向于选择拥有较多可能属性值的属性。为了弥补这一缺陷，一个称为增益率（gain ratio）的量度修正被广泛采用。增益率的获得考虑了属性分裂数据集后所产生的子节点的数量和规模，而忽略任何有关类别的信息。在图4-5的情形中，每个分支只含1个实例，所以分裂后的信息值为：

info([1，1，⋯，1]) = − 1/14 × log1/14 × 14

因为同样的分数1/14出现了14次，所以结果为log14或3.807位，这是一个非常高的信息值。因为分裂后所产生的信息值是指要确定各个实例应该被分配到哪个分支上所需要的位数，分支越多，这个信息值越大。增益率的计算方法是将原来的信息增益，在这个例子中是0.940，除以这个属性的信息值3.807，得到标识码属性的增益率0.247。

再返回图4-2天气数据的树桩，属性阴晴将数据集分裂成3个子集，规模分别为5、4和5，因此不考虑子集中所包含的类别，产生一个内在的信息值：

info([5，4，5])=1.577

可以看出，越是高度分支的属性，这个内在的信息值也越大，正如在假定的标识码属性上得到的信息值。信息增益可以通过将其除以内在的信息值，从而获得增益率的方法来进行修正。

表4-7总结了对图4-2所示树桩的计算结果。属性阴晴的结果依然排在首位，而属性湿度以一个更为接近的值排在第二位，因为它将数据集分裂成2个子集而不是3个。在这个例子中，假定的标识码属性以0.247的增益率，仍然是四个属性中的首选。然而，它的优势已经大大降低。在实际的开发过程中，可以采用一个特别的测试来防止在这类无用属性上的分裂。

表4-7　对图4-2所示树桩的增益率计算

阴　晴		温　度		湿　度		刮　风	
信息量:	0.693	信息量:	0.911	信息量:	0.788	信息量:	0.892
增益: 0.940-0.693	0.247	增益: 0.940-0.911	0.029	增益: 0.940-0.788	0.152	增益: 0.940-0.892	0.048
分裂信息量: info([5,4,5])	1.577	分裂信息量: info([4,6,4])	1.557	分裂信息量: info ([7,7])	1.000	分裂信息量: info([8,6])	0.985
增益率: 0.247/1.577	0.157	增益率: 0.029/1.557	0.019	增益率: 0.152/1	0.152	增益率: 0.048/0.985	0.049

104

不幸的是，在一些情况下增益率修正法补偿过度，会造成倾向于选择某个属性的原因，仅仅是因为这个属性的内在信息值比其他属性要小很多。一个标准的弥补方法是选择能够得到最大增益率的属性，而且那个属性的信息增益至少要等于所有属性的信息增益的平均值。

4.3.3 讨论

用分治法解决决策树归纳问题，有时称为自上而下的决策树归纳法（top-down induction of decision trees），由澳大利亚悉尼大学的J. Ross Quinlan开发，并经过了多年的改进。尽管其他机器学习的研究人员也在研究相似的方法，但是Quinlan的研究成果总是处于决策树归纳技术的最前沿。前面所描述的使用信息增益标准，从本质上看与称为ID3的方法是一致的。使用增益率是多年来用于改进ID3的方法之一。Quinlan认为它是一个广泛适用于不同情况的稳定方案。尽管它是一个强壮且实用的方案，但是它牺牲了一些信息增益标准的部分清晰、优雅的理论动机。

在称为C4.5的决策树归纳的一个实用且有影响力的系统中，积累了一系列改进ID3的方法。这些改进措施包括处理数值属性、残缺值、干扰数据以及由树产生规则的方法，有关内容将在第6.1节中描述。

4.4 覆盖算法：建立规则

如上所述，决策树算法是基于分治法来解决分类问题的。它们自上而下，在每一个阶段寻找一个能将实例按类别分隔的最佳属性，然后对分隔所得的子问题进行递归处理。这种方法产生出一个决策树，如果有必要可以将它转换成一个分类规则集，尽管要从决策树中产生有效规则，但这个转换过程并不简单。

另一个方法是依次取出每个类，寻找一个方法使之能覆盖所有属于这个类别的实例，同时能剔除不属于这个类别的实例。这种方法称为覆盖（covering）方法，因为在每一个阶段都要找出一个能够“覆盖”部分实例的规则。覆盖法的自身特性决定了它将产生一个规则集而不是一个决策树。

可以在二维的实例空间上观察覆盖法，如图4-6a所示。首先产生一个覆盖*a*类实例的规则。 105
作为规则中的第一个测试，如中间图所示那样，垂直分割属性空间。由此给出一个起始规则：

```
If x > 1.2 then class = a
```

然而，这个规则同时覆盖了很多*a*类和*b*类的实例，所以在规则中增加一个新的测试，如第3个图所示，在水平方向进一步对实例空间进行分割。

```
If x > 1.2 and y > 2.6 then class = a
```

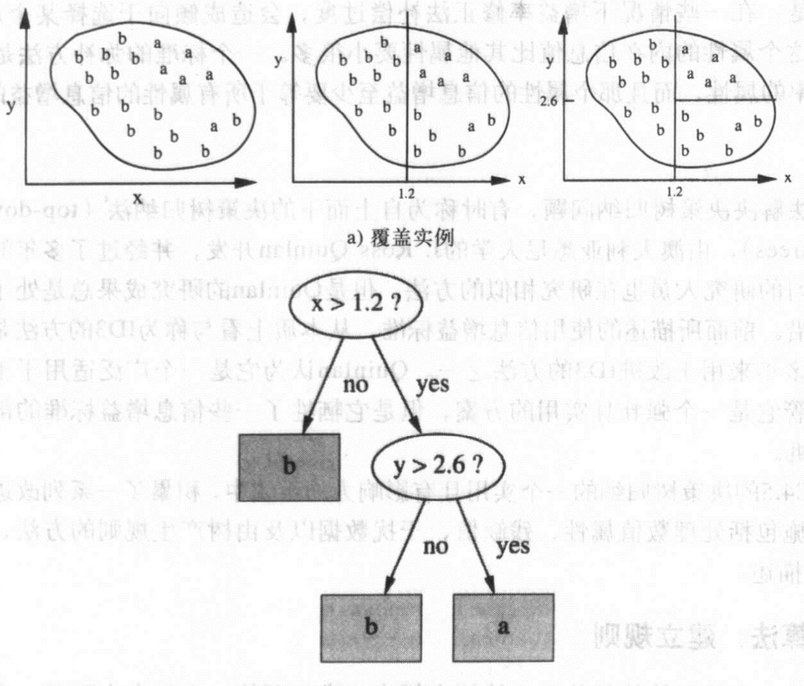
这个规则覆盖了除了一个*a*类实例以外的其他所有a类的实例。也许可以就此结束，但是如果感觉有必要覆盖最后一个*a*类实例，也许需要添加另一个规则：

```
If x > 1.4 and y < 2.4 then class = a
```

同样的过程可以产生出覆盖*b*类实例的2个规则是：

```
If x < 1.2 then class = b
If x > 1.2 and y ≤ 2.6 then class = b
```

同样，有一个*a*类实例被错误地包含进来。如果有必要去除它，就必须在第二个规则中增加更多的测试，并且需要再添加额外规则来覆盖被新测试排除在外的*b*类实例。



a) 覆盖实例

b) 同样问题的决策树

106 图4-6 覆盖算法

4.4.1 规则与树

对于同一个数据，自上而下的分治算法，至少从表面上看，与覆盖算法是很相似的。它也可能使用x属性对数据集进行首次分裂，也很可能在相同的位置$x = 1.2$处进行分裂。然而，覆盖算法仅考虑覆盖一个类别的实例，而由分治法产生的分裂则需要同时考虑两个类别，因为分治算法是建立一个能够应用于所有类别上的单一的概念描述。第二次分裂或许也在同样的位置，$y = 2.6$，从而产生如图4-6b所示的决策树。这个树与规则集完全对应，在这个例子中，覆盖法和分治算法本质上并没有什么不同。

但是在许多情况下，规则和树在表达的清楚明晰上存在差异。例如，当在第3.3节讨论重复子树问题时，曾经提到规则可以是对称的，而树必须首先选择一个属性进行分裂，这会导致树比一个等效的规则集大得多。另一个不同之处是，在多类的情况下，决策树分裂将考虑所有类别的情况，试图使分裂的纯度最大化，而规则建立法一次只集中处理一个类别，并不考虑其他类别上发生的情况。

4.4.2 一个简单的覆盖算法

覆盖算法向正在建设中的规则里添加测试，总是要尽力创建一个能获得最大正确率的规则。相反，分治算法是向正在建设中的树上添加测试，总是要尽力将不同类别最大程度地分开。这两种方法都牵涉到要寻找某个属性进行分裂的过程。但是两者寻找最佳属性的标准是不同的。分治算法，如ID3，选择一个属性能使信息增益最大化，而我们即将讨论的覆盖算法则要选择一个属性-值配对（attribute-value pair）能使期望类别概率达到最大化。

图4-7展示了一个实例空间，这个空间包含了所有的实例、一个部分创建完成的规则（partially constructed rule），和同样的规则在加入一个新条件后的情况。这个新的条件限制了规则的覆盖量：指导思想是要尽可能多地包含期望类别的实例，同时尽量不包含其他类别的实例。假设新的规则将总共覆盖t个实例，其中存在p个属于这个期望类别的实例，以及$t-p$个其他类别的实例，即规则所产生的错误。然后，选择新的条件使p/t最大化。

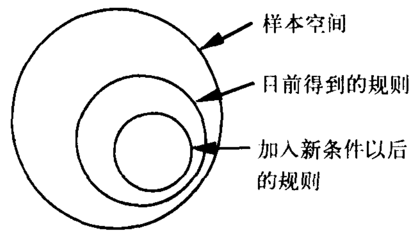


图4-7 覆盖算法操作过程中的实例空间

举个例子来帮助理解。这次改为使用表1-1的隐形眼镜数据。依次生成3个规则，分别覆盖3个类别（hard、soft和none）中的每一个。首先寻找一个规则：

```
If ? then recommendation = hard
```

对于未知条件“?”，存在9个选择：

```
age = young                                2/8
age = pre-presbyopic                       1/8
age = presbyopic                           1/8
spectacle prescription = myope             3/12
spectacle prescription = hypermetrope      1/12
astigmatism = no                           0/12
astigmatism = yes                          4/12
tear production rate = reduced             0/12
tear production rate = normal              4/12
```

右边的数值表示由这个条件选出的实例集中，“正确”实例的比例。这里，正确（correct）意味着建议（recommendation）使用硬的（hard）隐形眼镜。例如，由条件age=young选出8个实例，其中2个建议使用硬的隐形眼镜，所以第一个比例值是2/8。（为了能很好理解，最好看一下表1-1的隐形眼镜数据，并统计表中的记录。）选择最大一个比例值4/12，从上面列表里的第7个和最后一个之间任意选一个，建立规则：

```
If astigmatism = yes then recommendation = hard
```

107 ~ 108

这个规则并不非常准确，只从它所覆盖的12个实例中得到4个正确的实例，见表4-8。因此对它进行进一步修正：

```
If astigmatism = yes and ? then recommendation = hard
```

表4-8 *astigmatism = yes*的部分隐形眼镜数据

年龄	视力诊断	散光	泪流量	推荐镜片
young	myope	yes	reduced	none
young	myope	yes	normal	hard
young	hypermetrope	yes	reduced	none
young	hypermetrope	yes	normal	hard
pre-presbyopic	myope	yes	reduced	none
pre-presbyopic	myope	yes	normal	hard
pre-presbyopic	hypermetrope	yes	reduced	none
pre-presbyopic	hypermetrope	yes	normal	none
presbyopic	myope	yes	reduced	none
presbyopic	myope	yes	normal	hard
presbyopic	hypermetrope	yes	reduced	none
presbyopic	hypermetrope	yes	normal	none

未知条件“?”的可能性有7个，产生7个选择:

```
age = young                                  2/4
age = pre-presbyopic                         1/4
age = presbyopic                             1/4
spectacle prescription = myope               3/6
spectacle prescription = hypermetrope        1/6
tear production rate = reduced               0/6
tear production rate = normal                4/6
```

(再次统计表4-8中的记录。)很清楚，最后一个胜出，6个实例中有4个正确，相应的规则是:

```
If astigmatism = yes and tear production rate = normal
   then recommendation = hard
```

可以就此停止了吗? 也许。但是现在我们要找出丝毫不差的规则，而不管它们有多么复杂。表4-9所列的是到目前为止规则所能覆盖的实例。那么下一个可能的条件是:

```
age = young                                  2/2
age = pre-presbyopic                         1/2
age = presbyopic                             1/2
spectacle prescription = myope               3/3
spectacle prescription = hypermetrope        1/3
```

109 需要在第1和第4之间选择一个。到目前为止，都是将分数当作数值来处理，尽管这两个分数值相等（都为1），但是它们有不同的覆盖量：一个选择条件仅覆盖2个正确的实例，而另一个覆盖了3个。在同等条件下，总是选择拥有更大覆盖量的那个规则，所以最终的规则为:

```
If astigmatism = yes and tear production rate = normal
   and spectacle prescription = myope then recommendation = hard
```

表4-9　astigmatism = yes和tear production rate = normal的部分隐形眼镜数据

年　龄	视力诊断	散　光	泪流量	推荐镜片
young	myope	yes	normal	hard
young	hypermetrope	yes	normal	hard
pre-presbyopic	myope	yes	normal	hard
pre-presbyopic	hypermetrope	yes	normal	none
presbyopic	myope	yes	normal	hard
presbyopic	hypermetrope	yes	normal	none

这确实是在隐形眼镜问题上产生的规则之一。但是它仅包含4个建议使用硬的隐形眼镜情况中的3个。因此，从实例集中删除这3个实例，并且重新开始寻找另一种形式的规则:

```
If ? then recommendation = hard
```

按照同样的程序，将最终发现age=young是作为第一个条件的最佳选择。这个条件的覆盖量是7，7是因为有3个实例已经从原有的实例集中被删除了，总共还剩下21个实例。第2个条件的最佳选择是astigmatism=yes，选择的是1/3（实际上，还存在一个相同的比例值）; tear production rate=normal是第3个最佳选择，选择的是1/1。

```
If age = young and astigmatism = yes and
   tear production rate = normal then recommendation = hard
```

这个规则实际上覆盖了原始数据集中的3个实例，其中的2个已经被前面建立的规则覆盖了——但这也没有关系，因为这两个规则给出的建议是一样的。

现在所有的硬的隐形眼镜的实例都已经覆盖了，下一步是用相同的步骤生成软的隐形眼

镜的规则。最后生成none类别的规则。除非所寻找的规则集是带有缺省规则的，如果是这种情形，便不需要给最后的类别寻找明确的规则了。

以上的描述是用PRISM法来创建规则。它只建立正确或者“完美”的规则。它利用正确率公式*p*/*t*来衡量规则的成功率。任何低于100%正确率的规则都是“不正确”，因为低于100% 110 正确率的规则意味着将实例分配到并不是它所属的类别上。PRISM不断向每一个规则增加条件，直到规则达到完美：规则的正确率是100%。图4-8给出了这个算法的总结。最外面的循环在各个类上重复，依次为每个类别产生规则。注意在每次循环开始以前，需要将样本集重新初始化到完整的数据集状态。然后为那个类别建立规则，并从数据集中删除实例，直到没有这个类别的实例存在为止。每当开始创建规则时，要从一个空的规则开始（空规则覆盖了所有的样本），接着不断加入测试来限制规则，直到规则仅覆盖所期望的类别为止。在每一个阶段，选择最有希望的测试，就是使规则的正确率达到最大化的一个测试。最后，选择拥有最大覆盖量的测试来打破平局。

```
对于每个类别C
    将实例集初始化为E
    当E中含有类别C的实例时
        创建规则R，规则左边为空条件，预测类别为C
        循环工作，直到R完美（或者没有属性可使用了）
            对于每个在规则R中还未包含的属性A和属性值v
                考虑添加条件A = v到规则R的左边
                选择A和v，使正确率p/t最大化
                （选择最大覆盖量p的条件来打破平局）
        将A=v添加到规则R中
    将规则R所覆盖的实例从E中删除
```

图4-8　一个基本规则学习法的伪代码

4.4.3　规则与决策列

考虑为某个特定的类别产生的规则，也就是将图4-8所示算法中的最外层循环去掉。从规则产生的过程似乎可以很清楚地看到，这些规则打算要按照先后次序进行解释，就像是一个决策列，依次测试规则直到找到一个适用的规则，然后使用它。这是因为只要新的规则建立完成，由新规则所覆盖的实例将从实例集中被删除（图4-8倒数第3行代码）：因此后来生成的规则是为那些没有被这个规则所覆盖的实例设计的。然而，尽管表面看来规则应该依次被检 111 验，但这并不是必需的。考虑到后来为这个类别建立的规则都预测出同样的类别。这意味着它们以什么样的顺序执行是没有关系的：如果能够找到一个覆盖这个实例的规则，就可以预测出这个实例的类；或者找不到这样的一个规则，也就不可能预测出实例的类。

现在回到完整的算法上。依次考虑每一个类，并且产生出能够将属于这个类别的实例从其他类别的实例中识别出来的规则。为某个类别所建的规则和为其他类别所建的规则之间没有隐含的顺序关系。因此，产生出的规则能够以独立于顺序的方式执行。

正如在第3.3节所讨论的，顺序独立（order-independent）的规则似乎以各自独立的“知识”金块的行为方式，使规则更具有模块性，但同时也存在劣势，即当所采用的规则起冲突时，便不清楚应该如何处理了。用这种方法产生的规则，会使一个测试样本得到多个分类，

即它能够适用于不同类别上的规则。而其他的测试样本也可能不会得到任何分类。处理这种情况的一个简单的策略是在模糊事件上强制执行一个决策，从所预测出的类别中选择（与类别对应的）训练实例数量最多的那个类别，或者如果没有预测出的类别，那么选择在总体上拥有最多训练实例的类别。这种问题不会出现在决策列上，因为决策列需要按顺序解释，并且一旦一个规则适用就立刻停止解释过程：最后的一个缺省的附加规则能够保证任何测试实例都会有一个分类。采用一个稍微不同的方法，也许能为多类问题产生一个好的决策列。在第6.2节将涉及相关内容。

诸如PRISM之类的算法可以描述为割治算法（separate-and-conquer）：寻找一条能够覆盖许多同类实例的规则（去除不属于此类的实例），将这些已经被覆盖的实例从实例集中减除，因为这些实例已经被规则考虑到了，然后继续在剩下的实例上执行这个过程。这种方法和分治决策树方法是很妙的对照。减除（实例）的步骤大大提高了算法的效率，因为实例集的规模在操作过程中不断地缩减。

4.5 挖掘关联规则

关联规则（association rules）与分类规则相近似，可以采用同样的方法找出关联规则，方法是对每个可能出现在规则右边的表达式，执行一个分治的规则归纳过程。不但任何属性都可以伴随着任何可能的属性值出现在右边，而且一个单独的关联规则经常能够预测出不止一个属性的值。要找出这些规则，必须对在右边的每一种可能的属性组合（combination of
112 attributes），用每种可能的属性值的组合，执行一次规则归纳过程。从中将产生出数量庞大的关联规则，因此必须根据它们的覆盖量（应用规则预测正确的实例数量）和正确率（同样的数值但表示为它在规则应用所涉的全部实例中占据的比例）对其进行修剪。但是这种方法非常不可行。（注意，正如第3.4节所述，覆盖量经常称为支持（support），正确率（accuracy）也经常称为置信度（confidence）。）

这里需要说明，我们仅对拥有高覆盖量的关联规则感兴趣。暂时忽略一个规则左右两边的不同，只寻找能够达到预定最小覆盖量的属性-值配对的组合。它们（这些组合）称为项集（item set）：一个属性-值配对就是一个项（item）。这个术语出自购物篮分析，项是购物车里的商品，超市经理需要寻找购物篮里物品之间的关联。

4.5.1 项集

表4-10的第1列列出了表1-2天气数据的单独项，并在右边给出这个项在数据集上出现的次数。它们是单项集。下一步要建立二项集，方法是将单独项进行配对。当然，没有理由要建立一个包含两个相同属性，但是不同属性值的二项集（如阴晴= sunny和阴晴= overcast），因为它不可能出现在任何一个真正的实例上。

假设要寻找最小覆盖量为2的关联规则，就要去除那些覆盖实例的个数小于2的项集。因此将剩下47个二项集，表4-10的第2列显示了其中的部分二项集以及二项集在数据集中出现的次数。接下来要建立三项集，将有39个三项集的覆盖量为2或大于2。在这个数据集上，存在6个四项集，没有五项集，一个覆盖量是2或大于2的五项集只可能与一个重复的实例相对应。例如，表中的前两行显示阴晴= sunny有5天，其中温度= hot有2天。实际上，这两天又都是湿度=high和玩=no。

表4-10 覆盖量等于2或大于2的天气数据的项集

	单项集	二项集	三项集	四项集
1	outlook=sunny (5)	outlook = sunny temperature =hot (2)	outlook = sunny temperature = hot humidity = high (2)	outlook = sunny temperature = hot humidity = high play = no (2)
2	outlook=overcast (4)	outlook = sunny temperature = hot (2)	outlook = sunny temperature = hot play = no (2)	outlook = sunny humidity = high windy = false play = no (2)
3	outlook=rainy (5)	outlook = sunny humidity = normal (2)	outlook = sunny humidity = normal play = yes (2)	outlook = overcast temperature = hot windy = false play = yes (2)
4	temperature = cool (4)	outlook = sunny humidity = high (3)	outlook = sunny humidity = high windy = false (2)	outlook = rainy temperature = mild windy = false play = yes (2)
5	temperature = mild (6)	outlook = sunny windy = true (2)	outlook = sunny humidity = high play = no (3)	outlook = rainy humidity = normal windy = false play = yes (2)
6	temperature = hot (4)	outlook = sunny windy = false (3)	outlook = sunny windy = false play = no (2)	temperature = cool humidity = normal windy = false play=yes (2)
7	humidity = normal (7)	outlook = sunny play = yes (2)	outlook = overcast temperature = hot windy = false (2)	
8	humidity = high (7)	outlook = sunny play = no (3)	outlook = overcast temperature = hot play = yes (2)	
9	windy = true (6)	outlook = overcast temperature = hot (2)	outlook = overcast humidity = normal play = yes (2)	
10	windy = false (8)	outlook = overcast humidity = normal (2)	outlook = overcast humidity = high play = yes (2)	
11	play = yes (9)	outlook = overcast humidity = high (2)	outlook = overcast windy = true play = yes (2)	
12	play = no (5)	outlook = overcast windy = true (2)	outlook = overcast windy = false play = yes (2)	
13		outlook = overcast windy = false (2)	outlook = rainy temperature = cool humidity = normal (2)	
…		…	…	

（续）

	单项集	二项集	三项集	四项集
38		humidity = normal windy = false (4)	humidity = normal windy = false play = yes (4)	
39		humidity = normal play = yes (6)	humidity = high windy = false play = no (2)	
40		humidity = high windy = true (3)		
…		…		
47		windy = false play = no (2)		

4.5.2 关联规则

下面将讨论如何有效地建立项集，但是需要首先介绍如何将项集转换成规则。一旦拥有规定覆盖量的项集建立完毕，紧接着就要分别将项集转换成至少拥有指定最小正确率的规则或者是规则集。有一些项集将会产生多个规则，而其他的一些项集也许根本产生不出任何规则。例如，一个覆盖量是4的三项集（表4-10第38行）

```
humidity = normal, windy = false, play = yes
```

这个三项集将产生7个潜在的规则：

```
If humidity = normal and windy = false then play = yes          4/4
If humidity = normal and play = yes then windy = false          4/6
If windy = false and play = yes then humidity = normal          4/6
If humidity = normal then windy = false and play = yes          4/7
If windy = false then humidity = normal and play = yes          4/8
If play = yes then humidity = normal and windy = false          4/9
If - then humidity = normal and windy = false and play = yes    4/12
```

右边的数值是所有3个条件都满足时的实例数量，即覆盖量除以先决条件满足时的实例数量。用分数形式表示规则正确时所对应的实例百分率，也就是规则的正确率。假设指定的最小正确率为100%，那么只能把第1个规则纳入最后的规则集。从表4-10中寻找先决条件表达式可以获得分数的分母（尽管部分未在表中列出）。上面的最后一条规则不存在先决条件，它的分母是数据集中实例的总数。

表4-11是天气数据的最终规则集，规则集的最小覆盖量是2，最小正确率是100%，以覆盖量排序。共有58条规则，其中3条规则的覆盖量是4，5条规则的覆盖量是3，50条的覆盖量为2。
113~115 只有7条规则的结论含两个条件，没有一条规则的结论是含两个以上条件的。第一条规则从上面讨论的项集中得到的。有时同一个项集里会产生出几条规则。例如，规则9、10和11是从表4-10第6行的四项集上产生的：

```
temperature = cool, humidity = normal, windy = false, play = yes
```

它的覆盖量是2。这个四项集的3个子集拥有的覆盖量也是2：

```
temperature = cool, windy = false
temperature = cool, humidity = normal, windy = false
temperature = cool, windy = false, play = yes
```

因此它们产生出规则9、10和11，这3条规则都拥有100%的正确率（在训练数据集上）。

表4-11 天气数据的关联规则

	关联规则			覆盖量	正确率
1	humidity = normal windy = false	⇒	play = yes	4	100%
2	temperature = cool	⇒	humidity = normal	4	100%
3	outlook = overcast	⇒	play = yes	4	100%
4	temperature = cool play = yes	⇒	humidity = normal	3	100%
5	outlook = rainy windy = false	⇒	play = yes	3	100%
6	outlook = rainy play = yes	⇒	windy = false	3	100%
7	outlook = sunny humidity = high	⇒	play = no	3	100%
8	outlook = sunny play = no	⇒	humidity = high	3	100%
9	temperature = cool windy = false	⇒	humidity = normal play = yes	2	100%
10	temperature = cool humidity = normal windy = false	⇒	play = yes	2	100%
11	temperature = cool windy = false play = yes	⇒	humidity = normal	2	100%
12	outlook = rainy humidity = normal windy = false	⇒	play = yes	2	100%
13	outlook = rainy humidity = normal play = yes	⇒	windy = false	2	100%
14	outlook = rainy temperature = mild windy = false	⇒	play = yes	2	100%
15	outlook = rainy temperature = mild play = yes	⇒	windy = false	2	100%
16	temperature = mild windy = false play = yes	⇒	outlook = rainy	2	100%
17	outlook = overcast temperature = hot	⇒	windy = false play = yes	2	100%
18	outlook = overcast windy = false	⇒	temperature = hot play = yes	2	100%
19	temperature = hot play = yes	⇒	outlook = overcast windy = false	2	100%
20	outlook = overcast temperature = hot windy = false	⇒	play = yes	2	100%
21	outlook = overcast temperature = hot play = yes	⇒	windy = false	2	100%
22	outlook = overcast windy = false play = yes	⇒	temperature = hot	2	100%
23	temperature = hot windy = false play = yes	⇒	outlook = overcast	2	100%
24	windy = false play = no	⇒	outlook = sunny humidity = high	2	100%
25	outlook = sunny humidity = high windy = false	⇒	play = no	2	100%
26	outlook = sunny windy = false play = no	⇒	humidity = high	2	100%
27	humidity = high windy = false play = no	⇒	outlook = sunny	2	100%
28	outlook = sunny temperature = hot	⇒	humidity = high play = no	2	100%
29	temperature = hot play = no	⇒	outlook = sunny humidity = high	2	100%
30	outlook = sunny temperature = hot humidity = high	⇒	play = no	2	100%
31	outlook = sunny temperature = hot play = no	⇒	humidity = high	2	100%
…	…			…	
58	outlook = sunny temperature = hot	⇒	humidity = high	2	100%

4.5.3 有效地建立规则

现在来详细考虑一个使产生的关联规则达到指定的最小覆盖量和正确率的算法。这个算法分两个阶段：首先产生达到指定最小覆盖量的项集，然后从每一个项集中找出能够达到指定最小正确率的规则。

第一步是产生所有能达到给定最小覆盖量的单项集（如表4-10第1列），然后使用单项集产生二项集（第2列）、三项集（第3列），等等。每一步操作都要对整个数据集访问一遍，统计每种项集的数量，访问结束后将合格的项集保存在一个散列表——一种标准的数据结构中，散列表中的各个元素能够被迅速找出。从单项集中产生出二项集的候选成员，然后再对数据集访问一遍，统计每个二项集的覆盖量；最终将那些小于最小覆盖量的二项集成员从散列表中删除。二项集成员只是将单项集成员成对取出的所有组合，因为只有构成一个二项集的两个单项集都达到最小覆盖量时，这个二项集才有可能达到最小覆盖量。这点具有通用性：如果三项集中的3个二项集都达到最小覆盖量，那么这个三项集才有可能达到最小覆盖量。对四项集也是如此。

举一个例子来解释如何产生项集的成员。假设有5个三项集：（A B C），（A B D），（A C D），（A C E）和（B C D），这里A是一个属性，如阴晴= sunny。那么将前两个成员合并，即
116~117 得到（A B C D），是一个四项集的成员，因为它的其他三项子集（A C D）和（B C D）的覆盖量都大于最小覆盖量。如果三项集是按字母进行排序的，就像这个例子所示的序列，那么仅需要考虑对前两个成员相同的三项集配对。例如，我们不考虑（A C D）和（B C D），因为（A B C D）也可以从（A B C）和（A B D）中产生，并且，如果它们两个不是三项集的成员，那么（A B C D）就不可能成为四项集的成员。所以，只能产生两对三项集的组合：一对是已经讨论过的（A B C）和（A B D），另一对是（A C D）和（A C E）。第二对产生一个四项集（A C D E），由于它的三项集没有全部达到最小覆盖量，所以它将被丢弃。散列表能够帮助做这种检查：只要依次将（四）项集中的各项移出，并检查剩余的三项集是否真正出现在散列表中。在这个例子中，只存在一个四项集候选成员（A B C D）。这个四项集是否真正拥有最小覆盖量，只能通过对数据集中的实例进行检查才能确认。

第二个处理阶段是取出每一个项集，从中产生出规则，并检查产生的规则是否拥有指定的最小正确率。如果规则的右边只有一个测试，任务将变得简单，只要依次将每个条件作为规则的结论来考虑，从项集中删除它，用完整项集的覆盖量除以结果子集的覆盖量（从散列表中获得）得到对应于规则的正确率。我们同样对结论中包含多个测试的关联规则感兴趣，将项集的每种子集放在右边，而将剩余的项作为前提条件放在左边，似乎必须对这些结果进行评估。

除非项集的规模较小，否则这种穷举法将使得计算强度过于密集，因为可能子集的数量会随着项集的规模成指数级增长。然而，还有一种更好的方法。观察在第3.4节论述关联规则时的一条规则：

```
If windy = false and play = no then outlook = sunny
                                    and humidity = high
```

如果这个双重结论（double-consequent）规则满足给定的最小覆盖量和正确率，那么从同样的项集中产生的两个单个结论（single-consequent）的规则也一定满足最小覆盖量和正确率：

```
If humidity = high and windy = false and play = no
   then outlook = sunny
If outlook = sunny and windy = false and play = no
   then humidity = high
```

反过来，如果其中某个单个结论的规则不能满足最小覆盖量和正确率，那么就没有必要考虑双重规则了。这给出了一个从单个结论规则建立双重结论的候选规则的方法，此法同样也适用于从双重结论规则中建立三重结论候选规则，依此类推。当然，必须由散列表检查每一个候选规则，观察它们的正确率是否真正大于指定的最小正确率。通常由这种方法检查的规则数要远远小于穷举法。有趣的是从含n个结论的规则中产生出含$(n+1)$个结论的候选规则的方法与前面讲述的从n项集中产生出$(n+1)$项集的候选项集是完全一样的方法。

4.5.4 讨论

通常需要在大型数据集中寻找关联规则，所以高效率的算法具有很高的应用价值。以上描述的算法对于每种不同规模的项集都要在数据集上访问一遍。有时候，由于数据集太大而 118 不能全部读入主内存，必须存储在硬盘上，因此值得考虑在一次访问操作中同时检查两个相邻规模的项集，以减少访问整个数据集的次数。例如，一旦二项集产生，在用实例集统计集合中真正存在的二项集数量之前，可以先从二项集中生成所有的三项集。尽管这种方法考虑的三项集的数量大于真正三项集的数量，但是访问整个数据集的次数将减少。

实际上，建立关联规则所需的计算量取决于指定的最小覆盖量。正确率的影响力较小，因为它并不会影响到访问整个数据集的次数。在很多情况下，需要在一个预定的最小正确率的标准上，选用一定数量，比如说50个能够达到最大可能覆盖量的规则。一种方法是从一个较高的覆盖量开始，然后逐渐降低，对于每一个覆盖量，重复执行整个寻找规则的算法，直到达到所要求的规则数量为止。

本书采用的表格输入格式，特别是基于这种格式的标准ARFF文件，在处理很多关联规则问题时效率很低。关联规则通常使用在属性具有二值(binary)特性时，也就是存在或不存在，并且，一个给定实例的大部分属性值不存在。这是在第2.4节讲述的一种稀疏数据的表达形式，相同的算法可以用于这类数据来寻找关联规则。

4.6 线性模型

我们所讨论过的决策树和规则在名词性属性上运作得非常自然，也可以扩展到数值属性上。把数值测试直接运用到决策树或规则归纳方法上，或者将数值属性事先离散成名词性属性的形式。在第6章和第7章将分别介绍如何做。然而，有一些方法可以很自然地运用于数值属性。首先看几个简单的方案，这些方案是更加复杂的机器学习方法的基础，那些复杂的学习方法将在以后讨论。

4.6.1 数值预测：线性回归

当结论或者类是数值，并且所有的属性值都是数值时，线性回归法是一种自然会考虑的技术。线性回归是统计学的一种常用方法。它的主导思想是利用预定的权值将属性进行线性组合来表示类别：

$$x = w_0 + w_1a_1 + w_2a_2 + \cdots + w_ka_k$$ 119

这里x是类，a_1，a_2，…，a_k是属性值，w_0，w_1，…，w_k是权值。

权值是从训练数据中计算出来的。这里的标记比较多，因为需要一种能表达每个训练实例的属性值的方法。第一个实例将有一个类，$x^{(1)}$，和属性值 $a_1^{(1)}, a_2^{(1)}, \cdots, a_k^{(1)}$，上标表示是第一个实例。此外，为了标注的方便，再假设一个额外属性a_0，它的值总是为1。

对第一个实例的类的预测值可以写成如下形式：

$$w_0 a_0^{(1)} + w_1 a_1^{(1)} + w_2 a_2^{(1)} + \cdots + w_k a_k^{(1)} = \sum_{j=0}^{k} w_j a_j^{(1)}$$

这是第一个实例类值的预测值，而不是真实值。我们感兴趣的是预测值和真实值的差异。线性回归法是选择总共$k+1$个系数w_j使所有训练实例上的预测值和真实值的差值的平方之和达到最小。假设有n个训练实例，第i个实例的上标是(i)。那么预测值和真实值的差值的平方之和为：

$$\sum_{i=1}^{n}\left(x^{(i)} - \sum_{j=0}^{k} w_j a_j^{(i)}\right)^2$$

括号里的表达式是第i个实例的真实类值和它的预测类值之差。这个平方和正是我们要通过选择适当的系数来达到最小化。

乍一看，这个公式有点复杂。但是，如果你有相关的数学基础，最小化技术便是直截了当的。如果给予的实例数量足够，简单地说，就是样本的数量要多于属性的数量，那么选择适合的权值使得差值的平方和达到最小并不难。尽管这个过程确实包含一个矩阵倒置的操作，但是有关的软件包已经开发出来，可以直接拿来使用。

一旦完成了数学计算，将得到一个基于训练数据的数值型的权值集合，可以用来预测新实例的类值。前面已经看过这样的例子，当查看CPU性能数据时，图3-7a给出了实际的数值型的权值。这个公式可用来对新的测试实例，进行CPU性能预测。

线性回归是一个出色的、简单的适用于数值预测的方法，在统计应用领域广泛使用了数
120 十年。当然，线性模型也存在缺陷。如果数据呈现出非线性关系，线性回归将会找到一条最适合的直线，“最适合”是指最小均方差（least mean squared difference）。这条线也许并不十分适合。然而，线性模型可以作为其他更为复杂的学习方法的基础。

4.6.2 线性分类：Logistic回归

线性回归法可以方便地应用于含有数值属性的分类问题。事实上任何回归技术，无论是线性的还是非线性的，都可以用来分类。技巧是对每一个类执行一个回归，使属于该类的训练实例的输出结果为1，而不属于该类的输出结果为0。结果得到该类的一个线性表达式。然后，对于一个给定的未知类的测试实例，计算每个线性表达式的值并选择其中最大的。这种方法有时称为多反馈线性回归（multiresponse linear regression）。

一种查看多反馈线性回归的方法是将线性表达式想像成与每个类对应的数值型的从属关系函数（membership function）。对于属于这个类别的实例，从属关系函数值为1，对于其他类别的实例，函数值为0。对于一个给出的新实例，计算新实例与各个类别的从属关系，从中选择（从属关系）最大的一个。

在实际应用中，多反馈线性回归通常能产生很好的结果。但是，也存在两个缺陷。第一，从属关系函数产生的不是概率值，因为从属关系值有可能落在0到1的范围以外。第二，最小

平方回归假设误差不但是统计上的独立，而且是呈现出具有相同标准差的正态分布，当多反馈线性回归用于分类问题时，明显地违背了这个假设，因为这时观察值仅呈现0和1。

一个相关的、称为logistic回归（logistic regression）的统计技术不存在这个问题。直接逼近0和1的方法会在超越目标时，出现非法的概率值，而logistic回归是在一个经转换的目标变量上建立一个线性模型。

首先假设只有两个类的情况。logistic回归将原始目标变量：

$$\Pr[1|a_1,\ a_2,\ \cdots,\ a_k]$$

这个无法用线性函数来正确地近似表达的变量，替换为：

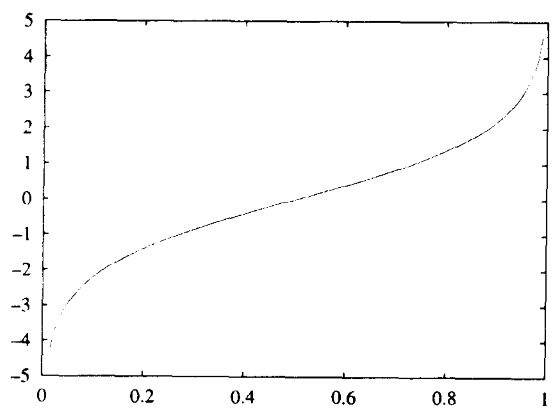
$$\log(\Pr[1|a_1,\ a_2,\ \cdots,\ a_k]/(1-\Pr[1|a_1,\ a_2,\ \cdots,\ a_k])$$

结果值将不再局限于0到1的区间，而是负无穷大和正无穷大之间的任何值。图4-9a是转换函数图，常称为对数变换（logit transformation）。

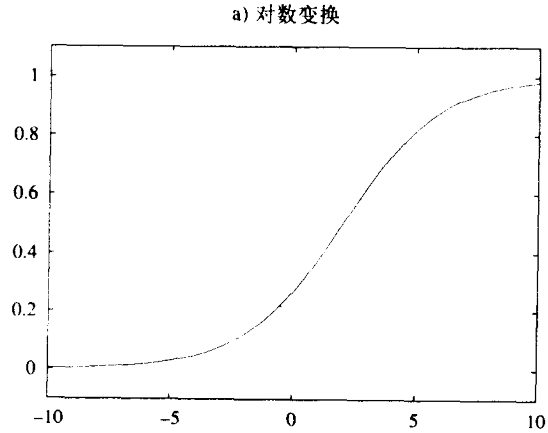
转换后的变量使用一个线性函数来近似，就像是由线性回归法所建立的函数。结果模型是：

$$\Pr[1|a_1,\ a_2,\ \cdots,\ a_k]=1/(1+\exp(-w_0-w_1a_1-\cdots-w_k\ a_k))$$ 121

这里权值为w。图4-9b展示了这个函数在一维数据空间上的一个例子，带有两个权值分别为$w_0=0.5$和$w_1=1$。



a) 对数变换



b) Logistic回归函数的例子

图4-9 Logistic回归

和线性回归一样，需要找出能与训练数据匹配得较好的权值。线性回归使用平方误差来
122 测量匹配的良好程度。在logistic回归里，则使用模型的对数似然（log-likelihood）。这便是：

$$\sum_{i=1}^{n}(1-x^{(i)})\log(1-\Pr[1\mid a_1^{(i)},a_2^{(i)},\cdots,a_k^{(i)}])+x^{(i)}\log(\Pr[1\mid a_1^{(i)},a_2^{(i)},\cdots,a_k^{(i)}])$$

这里$x^{(i)}$是0或者1。

应该选择能够使对数似然最大化的权值w_i。解决最大化问题的方法有几种。其中一种简单的方法是迭代地解决一系列加权最小平方回归问题，直到对数似然收敛于一个最大值处，通常在经过几次的迭代过程即可。

将logistic回归推广到多类问题，一个可能的方法是如前面多反馈线性回归里讲述的，为每个类别独立地形成logistic 回归（模型）。不幸的是，所得到的概率值之和不为1。为了获得适当的概率，有必要将用于每个类别的各体模型结合起来。这样将产生出一个联合优化问题，已经有一些解决方案能有效地处理这个问题。

一个概念更简单，并且非常通用的解决多类问题的方法是成对分类（pairwise classification）。为每一对类别建立一个分类器，仅使用属于这两个类别的实例。在一个未知测试实例上的结论是看测试实例在哪个类上能得到最多的投票。从分类误差来看，这种方案通常能产生正确的结论。它也能够通过应用一个称为成对合并（pairwise coupling）的方法产生概率估计，这种方法可以校正从不同分类器上产生的各自的概率估计。

如果有k个类，成对分类法总共建立$k(k-1)/2$个分类器。尽管看起来这个计算强度是不必要的，但是实际上，如果类分布均衡，那么成对分类法的处理速度至少和其他处理多类的方法一样快。原因是每一个成对的学习问题只考虑属于这两个类的实例。如果将n个实例平分到k个类上，分到每个成对的学习问题上的实例数量为$2n/k$。假如一个有n个实例的二类问题，机器学习算法需要花费与n秒成比例的时间去执行。那么成对分类法则需要与$k(k-1)/2\times 2n/k$秒成比例的时间，也就是与$(k-1)n$秒成正比。换句话说，这种方法与类别数量呈线性关系。如果学习算法需要花费更多时间，如与n^2成比例，那么成对分类法的优势就会更加明显。

线性函数处理分类问题的使用方法能够很容易地在实例空间上进行察看。二类问题的logistic回归决策边界是在预测概率为0.5处：

$$\Pr[1\mid a_1,a_2,\cdots,a_k]=1/(1+\exp(-w_0-w_1a_1-\cdots-w_ka_k))=0.5$$

123 它发生在

$$-w_0-w_1a_1-\cdots-w_ka_k=0$$

时。由于这是关于属性值的线性等式，所以边界是一个在实例空间上的线性平面，或称超平面（hyperplane）。很容易观察到不能由单个超平面分隔的实例点的集合，这些便是logistic 回归模型不能正确区分的实例。

多反馈线性回归也存在同样问题。每一个类获得一个从训练数据上计算出的权值向量。先着重讨论一对具体的类别。假如类1的权值向量是：

$$w_0^{(1)}+w_1^{(1)}a_1+w_2^{(1)}a_2+\cdots+w_k^{(1)}a_k$$

类2的权值向量是上标为2的同样的表达式。如果对于一个实例存在：

$$w_0^{(1)}+w_1^{(1)}a_1+\cdots+w_k^{(1)}a_k>w_0^{(2)}+w_1^{(2)}a_1+\cdots+w_k^{(2)}a_k$$

那么这个实例将被分配到类1而不是类2。换句话说，就是一个实例将被分配到类1的条件是：

$$(w_0^{(1)}-w_0^{(2)})+(w_1^{(1)}-w_1^{(2)})a_1+\cdots+(w_k^{(1)}-w_k^{(2)})a_k>0$$

这是一个关于属性值的线性不等式，所以每对类之间的边界是一个超平面。在执行成对分类时，也是这样。唯一的不同之处是两个类之间的边界由属于这两个类别的训练实例控制，而且不会受到其他类别实例的影响。

4.6.3 使用感知器的线性分类

logistic回归试图通过将训练数据的概率最大化的方法，产生出正确的概率估计。当然，正确的概率估计会产生出正确的分类。但是，如果模型的唯一目的只是预测类的标签，没有必要进行概率估计。一种不同的方法是学习一个超平面，能将属于不同类别的实例分开，假设只有两个类。如果使用一个超平面能够将数据完美地分成两组，那么就称该数据为可线性分隔（linearly separable）的数据。如果数据是可线性分隔的，便有一个非常简单的算法用于寻找一个分隔超平面。

这种算法称为感知器学习规则（perceptron learning rule）。在仔细研究它之前，再来看一下用于表示超平面的等式：

$$w_0a_0+w_1a_1+w_2a_2+\cdots+w_ka_k=0$$

这里，a_1，a_2，…，a_k 分别是属性的值，w_0，w_1，…，w_k 是定义超平面的权值。假设扩展每一个训练实例a_1，a_2，…，a_k 使它存在一个额外属性a_0，属性值始终为1（正如在线性回归里一样）。这个扩展属性称为偏差（bias），意味着在求总和时，不必包含一个额外的常量元素。如果所求出的和大于0，将它预测成第一个类，否则为第二个类。我们希望找出权值，那样训练数据就可以被超平面正确地分隔开。 124

图4-10a给出了为寻找一个分隔超平面的感知器学习规则。这个算法不断迭代直到找出一个完美的解决方案，但只有当数据中确实存在一个分隔超平面时，也就是当数据是可线性分隔时才能很好地工作。每次循环都要在所有训练实例上运行。如果遇到一个错分的实例，就要改变超平面的参数，让错分的实例更靠近超平面，或者甚至跃过超平面进入正确的一边。如果实例属于第一个类别，便将它的属性值加入权值向量，否则从权值向量中减去它的属性值。

现在解释为什么这样做。考虑一个属于第一类别的实例a加进来以后：

$$(w_0+a_0)\,a_0+(w_1+a_1)\,a_1+(w_2+a_2)\,a_2+\cdots+(w_k+a_k)\,a_k$$

这意味着a的输出所得的增加值为：

$$a_0\times a_0+a_1\times a_1+a_2\times a_2+\cdots+a_k\times a_k$$

这个数总是正数。所以超平面将向能使实例a获得肯定分类的正确方向移动。相反，如果一个实例属于第二个类，而被错分，那么这个实例的输出经过修改后将降低，同样是将超平面朝正确的方向移动。

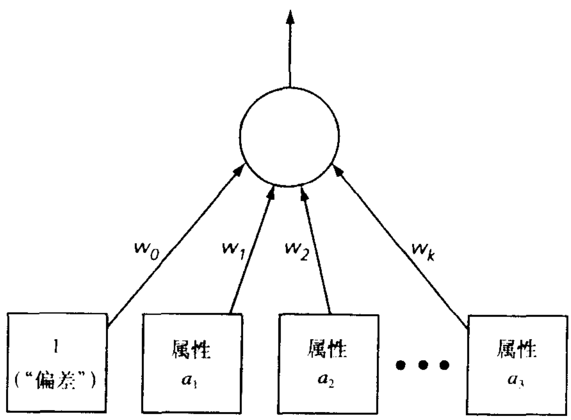
这种修正是递增的，会与先前的更新相抵触。然而，如果数据是可线性分隔的，那么在经过有限次的循环之后，算法将收敛。当然，如果数据不是可线性分隔的，算法将无法停止，所以当这种方法在实际运用中，需要强制设定一个循环次数的上限。

得到的超平面称为一个感知器，它是神经网络的前辈（在第6.3节将介绍神经网络）。图

4-10b是将感知器用包含节点和加权边的图型来表示，形象地称它为一个“神经”的“网络”。它有两层节点：输入和输出。输入层上每个节点代表一个属性，加上一个总是设置为1的额外节点。输出层仅有一个节点构成。每一个在输入层上的节点都被连接到输出层。这些连接是加权的，权值是由感知器学习规则找到的数值。

将所有权值设为0
循环操作，直到所有训练实例被正确分类
　　对于训练集中的每个实例I
　　　　如果实例I 被感知器错误分类
　　　　　　如果实例I属于第1个类别，则将其加入权值向量
　　　　　　否则从权值向量中将其减去

a) 学习规则



b) 以一个神经网络形式表达

图4-10　感知器

当一个实例放置在感知器上时，实例的属性值将“击活”输入层。属性值分别与权值相乘，并且在输出节点上求和。如果经加权的属性值之和大于0，那么输出结果为1，表示实例属于第一类；否则输出结果为－1，表示实例属于第二类。

4.6.4　使用Winnow的线性分类

感知器并非保证为可线性分隔的数据找到分隔超平面的唯一方法。对于二值（binary）属性的数据集，有一种处理方法称为Winnow算法，见图4-11a。两种算法的结构非常相似。和感知器一样，当出现错分的实例时，Winnow才更新权值向量。它是错误驱动（mistake driven）型的。

125～126 两种方法的不同之处在于如何更新权值。感知器规则应用一个加法机构，通过加上（或者减去）实例的属性向量来修改权值向量。Winnow采用乘法更新权值向量，将权值乘以用户指定的参数α（或者α的倒数）来分别地修改权值。属性a_i 的值为0或者1，因为处理的是二值数据。如果属性值为0，权值不会改变，因为它们没有参与决策。否则，如果属性帮助做出了一个正确的决策，那么乘数为α，如果没有帮助做出正确的决策，那么乘数为$1/a$。

```
当存在错误分类实例时
    对于每个实例a循环操作
        使用当前的权值对实例a进行分类
        如果预测类别不正确
            如果a属于第一个类
                对于每个属性值a_i，当a_i为1时将w_i乘以α
                （如果a_i为0，w_i保持不变）
            否则
                对于每个属性值a_i，当a_i为1时将w_i除以α
                （如果a_i为0，w_i保持不变）
```

a) 不平衡的Winnow算法

```
当存在错误分类实例时，
    对于每个实例a循环操作
        使用当前的权值对实例a进行分类
        如果预测类别不正确
            如果a属于第一个类
                对于每个属性值a_i，且a_i为1时
                    将w_i^+乘以α
                    将w_i^-除以α
                （如果a_i为0，w_i^+和w_i^-保持不变）
        否则
            对于每个属性值a_i，且a_i为1时
                    将w_i^-乘以α
                    将w_i^+除以α
            （如果a_i为0，w_i^+和w_i^-保持不变）
```

b) 平衡的Winnow算法

图4-11 Winnow算法

另一个不同是Winnow在线性函数中的阈值也是一个用户定义的参数。我们称这个阈值为θ，当且仅当满足以下条件时，才将这个实例分配到类1上：

$$w_0a_0 + w_1a_1 + w_2a_2 + \cdots + w_ka_k > \theta$$

乘数α需要大于1。在开始的时候将w_i设置成常量。

以上描述的算法不允许有负的权值，要看具体的应用领域，这可能成为一种缺点。然而，还有另一个版本称为平衡的Winnow（Balanced Winnow），平衡的Winnow允许使用负的权值。这个版本包含两个权值向量，每个类对应一个权值向量。如果一个实例满足以下条件，它将被分到类1中。图4-11b是平衡的Winnow算法：

$$(w_0^+ - w_0^-)a_0 + (w_1^+ - w_1^-)a_1 + \cdots + (w_k^+ - w_k^-)a_k > \theta$$

Winnow算法是对于跟踪数据集上的相关属性非常有效的方法，为此称为有效属性（attribute-efficient）学习器。如果一个数据集存在很多（二值）属性，并且其中的大部分属性不相关的，那么Winnow也许是一个好的候选算法。Winnow和感知器算法一样可以用于实时设置（online setting），在实时设置的情况下，新实例连续不断地到来，而每当有新的实例到达时，这两个算法能递增地更新它们的假定。

4.7 基于实例的学习

在基于实例的学习中，训练样本被一字不差地保存，并且使用一个距离函数来判定训练集中的哪个实例与一个未知的测试实例最靠近。一旦找到最靠近的训练实例，那么最靠近实例所属的类就被预测为测试实例的类。剩下的唯一问题就是定义距离函数，它并不十分困难，尤其是当属性为数值属性时。

4.7.1 距离函数

尽管存在其他可能的选择，但是大部分基于实例的学习方法使用欧几里得距离函数。属性值为 $a_1^{(1)}, a_2^{(1)}, \cdots, a_k^{(1)}$ （k是属性的个数）的实例与另一个属性值为 $a_1^{(2)}, a_2^{(2)}, \cdots, a_k^{(2)}$ 的实例之间
127~128 的距离定义为：

$$\sqrt{(a_1^{(1)}-a_1^{(2)})^2+(a_2^{(1)}-a_2^{(2)})^2+\cdots+(a_k^{(1)}-a_k^{(2)})^2}$$

比较距离时，不必计算平方根，直接使用平方之和进行比较。欧几里得距离可以由曼哈顿（Manhattan）或者city-block距离度量来替代，曼哈顿和city-block不是计算属性值差值的平方，而是将差值相加（取绝对值以后）。其他方法采用指数大于2的形式。更高的指数增加了大差异的影响力而削弱了小差异的影响力。通常欧几里得距离公式是一个很好的折中方法。在一些特殊场合，其他的距离度量法也许更为适合。关键是要思考真实的距离以及二者之间以某个具体距离分隔开意味着什么？又譬如，这个距离的两倍又意味着什么？

不同的属性用不同的尺度量度，如果直接使用欧几里得公式，某些属性的结果可能被另外一些使用较大量度尺寸的属性完全削弱。所以，通常需要将所有属性值正常化，在0和1之间。通过计算：

$$a_i = \frac{v_i - \min v_i}{\max v_i - \min v_i}$$

这里，v_i 是属性i的真实值，最大和最小属性值是从训练集的所有实例中获得的。

这些公式隐含的假设为数值属性。这里，两个值之间的差就是它们之间的数值差异，将这个差值平方以后再相加得到距离函数。对于名词性属性，属性值是符号值而不是数值，两个不同的名词性属性值之差常认为是1，如果名词性属性值相同，它们的差值为0。这里无需量度尺寸，因为只使用1和0。

一个通用的处理残缺值的方法如下。对于名词性属性，假设残缺属性值与其他属性值的差别达到最大值。因此如果两个属性值中的一个或者两个都残缺，或者如果两个属性值不同，那么它们之间的差值为1；只有两个属性值都不残缺，并且相同时，它们之间的差值才为0。对于数值属性，两个残缺值之差也为1。但是，如果仅有一个属性值残缺，那么它们的差值是另一个值的正常化值，或是1减去那个正常化值，取两者中较大的那个。这意味着如果属性值残缺，差值将会达到可能的最大差值。

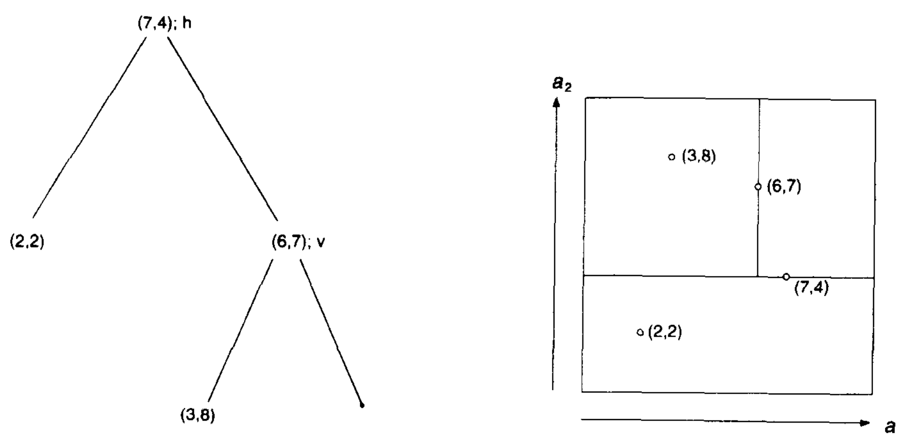
4.7.2 有效寻找最近邻

尽管基于实例的学习方法不但简单而且很有效果，但是通常速度很慢。一种显而易见、用于寻找哪个训练集成员最靠近类未知的测试实例的方法便是，计算训练集里的每一个训练
129 实例到测试实例的距离，并选择距离最小的那一个。这个过程与训练实例的数量成线性关系，

换句话说，就是做一个单独的预测所花费的时间与训练实例的数量成比例关系。处理整个测试集所花费的时间与训练集实例数量和测试集实例数量的乘积成比例关系。

以树的形式表示训练实例集能更加有效地找出最近邻实例，尽管怎样用树来表示并不十分明显。其中一种适合的树结构是*k*D树。*k*D树是一个二叉树，它用一个超平面将输入实例空间分隔开，然后再将每一个部分递归地进行分裂。在二维数据空间上，所有的分裂都与一个轴平行或者垂直。数据结构称为kD树，是因为它将一系列的数据点存储在*k*维空间，*k*是属性的数量。

图4-12a展示了$k = 2$的小例子，图4-12b显示了4个训练实例，以及构成树的超平面。注意这些超平面不是决策边界，分类决策将由稍后介绍的最近邻基础上做出。第一次分裂是水平分裂(*h*)，分裂点（7,4）是树的根节点。左支将不再进一步分裂，它包含了一个点（2，2），是树的叶子。右支在点（6，7）处进行垂直分裂（*v*）。它的右支为空，左支包含一个点（3，8）。如该例所示，每个区域只有一个点，或者没有点。树的同胞（sibling）分支，例如，图4-12a中根节点的两个子支，并不一定要发展到相同的深度。训练集中的每一个点与树的一个节点相对应，最多达一半的节点是叶节点。



a）树　　b）实例和分裂

图4-12　含4个训练实例的*k*D树 130

如何为一个数据集建一个*k*D树？当新的训练实例加入时，*k*D树能够进行有效的更新吗？*k*D树又是如何提升最近邻计算速度的？首先看最后一个问题。

为了找到一个给定目标点的最近邻，需要从树的根节点开始向下沿树找出目标点所在的区域。图4-13是一个近似于图4-12b的实例空间，只是多几个实例并增加了一条边界。目标点不是树上的实例中的一个，图中用了一个星标出。目标点所在区域的叶节点涂成黑色。正如该例所示，叶节点不一定是目标点的最近邻，但是，这是寻找最近点的很好的首次尝试。值得注意的是，任何更近的近邻点必须落在更近的地方，例如落在图4-13的虚线圆范围内。为了确定是否存在一个更近的近邻，首先检查叶节点的同胞节点是否有可能存在一个更近的近邻。黑色节点的同胞节点是图4-13有阴影的部分，但是虚线圆并没有与之相交，所以同胞节点内不可能包含更近的近邻。然后回溯到父节点，并检查父节点的同胞节点，父节点的同胞节点覆盖了所有横线以上的区域。在这个例子中，必须对个区域做进一步研究，因为这个区域与当前的最佳圆

相交。首先找出它的子节点（即初始点的两个叔辈节点），检查它们是否和圆相交（左边那个不相交，而右边那个与圆相交），并由此向下寻找是否存在一个更近点（存在）。

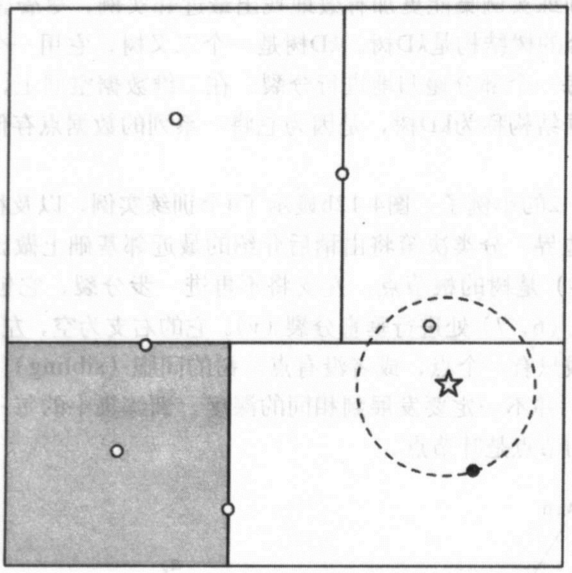


图4-13 使用kD树寻找星的最近邻

在典型的案例中，这个算法比考察所有点来寻找最近邻的方法快很多。寻找一个初始的近似最近邻点——图4-13中的黑色点，与树的深度密切相关，树的深度由树的节点个数取对数，即$\log_2 n$。回溯并检查是否存在最近邻的工作量有一小部分取决于树，另一部分取决于初始近似点的好坏程度。但是对于一个结构良好的树来说，它的节点近似于方形，而不是瘦长的矩形，这部分工作量也是取决于节点个数的对数。

如何才能在一个训练样本集上创建一棵好树？关键问题归结为对要分裂的第一个训练实例的选择以及分裂的方向。一旦完成这项工作，就可以在初始分裂所生成的每个子节点上递归应用相同的方法来完成建树过程。

为了给分隔寻找一个好的方向，需要分别计算数据点在每个轴向上的方差，选择最大方差所对应的轴，然后建立一个与该轴垂直的分隔超平面。为了给分隔超平面找到一个好位置，要找出位于那个轴向上的中间值，并选择与之相对应的点。这将使分隔面垂直于（数据）散布范围最广的方向，让每一边都拥有一半的数据点。这种方法将产生一个平衡的树。为了避免出现长条形区域，最好能使连续的分裂在不同轴向上。因为每个阶段都选择方差最大的轴向，因此很有可能满足这点。但是，如果数据点的分布非常不均衡，采用选择中间值的方法也许会在同一个方向上产生多次后续分隔，从而产生瘦长形的超矩形（hyperrectangle）。一个更好的解决方法是计算平均值而不是中间值，并使用最接近平均值的点。由此产生的树也不是完美的平衡，但是它的区域趋向于方形，因为这种方法增加了在不同方向上产生后续分隔

131 的机会。

与其他大部分机器学习方法相比，基于实例学习的一个优势是新的实例可以在任何时候加入到训练集里。在使用kD树时，为了保持这个优势，需要用新的数据点不断地更新这棵树。判断哪个叶节点包含了新的数据点，并且找出叶节点的超矩形。如果超矩形为空，就将新数

据点放置在那里。否则，分裂超矩形，分裂在最长的尺寸边上进行，以保持方形。这种简单的探索式方法并不能保证在加入一系列点以后，树依然会维持平衡，也不能保证为搜索最近邻塑造良好的超矩形。有时从头开始重建树不失为一个良策。例如，当树的深度达到最可能的深度值的两倍时。

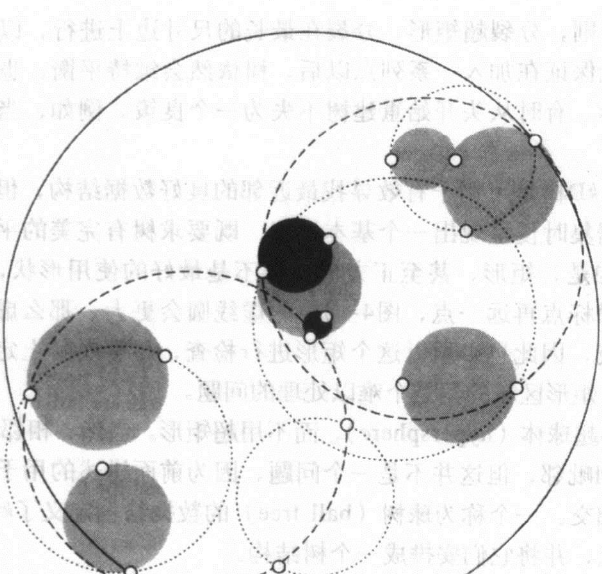
我们已经看到，kD树是可用于有效寻找最近邻的良好数据结构，但是，并不完美。当处理不均匀分布的数据集时便呈现出一个基本冲突：既要求树有完美的平衡结构，又要求区域近似方形。更重要的是，矩形，甚至正方形，都不是最好的使用形状，原因是它们都有角。如果黑色的实例离目标点再远一点，图4-13中的虚线圆会更大，那么虚线圆将有可能与左上方矩形的右下角相交，因此也必须对这个矩形进行检查，尽管实际上定义这个矩形的训练实例离这个方角很远。矩形区域的角是个难以处理的问题。 132

解决方案？使用超球体（hypersphere），而不用超矩形。当然，相邻的球体可能相互重叠，而矩形却可以彼此相毗邻，但这并不是一个问题，因为前面讲述的用于kD树的最近邻算法并不需要区域之间不相交。一个称为球树（ball tree）的数据结构定义了k维超球体（“球”），它覆盖了所有的数据点，并将它们安排成一个树结构。

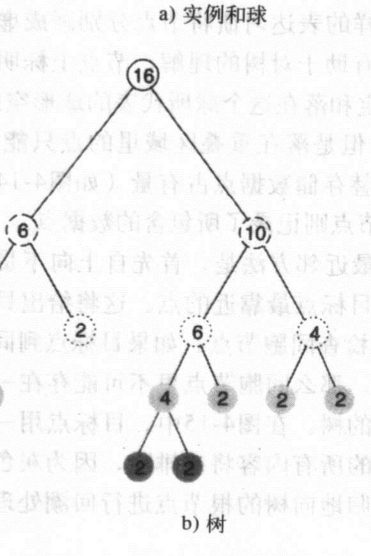
图4-14a展示了二维空间上的16个实例，被由重叠圆组成的图案所覆盖。图4-14b是由这些圆形成的树。在树的不同层上的圆用不同形式的虚线画出，更小的圆被打上灰色的阴影。树的每个节点代表一个球，采用同样的表达习惯将节点分别画成虚线或者打上阴影，这样就能清楚辨别出球属于哪一层。为了有助于对树的理解，节点上标明了数字以显示那个球里数据点的个数。注意：这个数字不一定和落在这个球所代表的球形空间区域里的数据点数量一致。在每一层上的区域有时会重叠，但是落在重叠区域里的点只能被分配到重叠球中的一个上（在图中看不出到底是哪个）。代替存储数据点占有量（如图4-14b所示），真实球树的节点上存储了球的中心点和半径，在叶节点则记录了所包含的数据点。

使用球树找出给定目标点的最近邻方法是，首先自上向下贯穿整棵树找出包含目标点所在的叶子，并在这个球里找出与目标点最靠近的点。这将给出目标点距离它的最近邻点的一个上限值。然后，和kD树一样，检查同胞节点。如果目标点到同胞节点中心的距离超过同胞节点的半径与当前的上限值之和，那么同胞节点里不可能存在一个更近的点；否则的话，必须进一步检查位于同胞节点以下的树。在图4-15中，目标点用一个星表示，黑色点是当前已知的目标点的最近邻。灰色球里的所有内容将被排除，因为灰色球的中心点离得太远，所以它不可能包含一个更近的点。递归地向树的根节点进行回溯处理，检查所有可能包含一个更近于当前上限值的点的球。

球树是自上而下地建立，和kD树一样，根本问题是要找到一个好的方法将包含数据点集的球分裂成两个。在实践中，不必等到叶子节点只有两个数据点时才停止，可以采用和kD树一样的方法，一旦节点上的数据点达到预先设置的最小数量时，便可提前停止建树过程。这里有一个可行的分裂方法。从球中选择一个离球的中心最远的点，然后选择第二个点离第一个点最远。将球中所有的点分配到离这两个聚类中心最近的一个上，然后计算每个聚类的中心，以及聚类能够包含它所有数据点所需的最小半径。这种方法的优点是分裂一个包含n个数据点的球的成本只是随n呈线性增加。其他更好的算法会产生出更紧凑的球，但是需要的计算量更大。这里将不再继续讨论复杂的算法，用于创建球树，或者用于新的实例加入时，对球树进行递增更新。 133



a) 实例和球



b) 树

图4-14　16个训练实例的球树

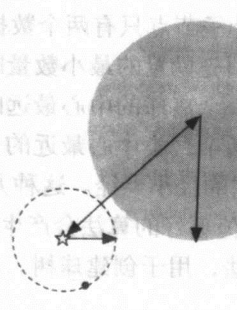


图4-15　根据目标点（星）和目标点当前最近邻排除整个球（灰色）

4.7.3 讨论

基于实例的最近邻学习方法不但简单，而且通常运作很好。在前面描述的方法中，每一个属性在决策上具有相同的影响力，就像朴素贝叶斯方法一样。另一个问题是数据库很容易受干扰样本破坏。一个解决方案是采用k最近邻法，找出固定的、小的、k个最近邻，如5个，让它们通过简单的投票方法（少数服从多数）共同决定测试实例的类别。（注意：前面曾使用k来代表属性的个数，这里的k是一个不同的、独立的使用方法。）另一种增强数据库抵抗干扰数据的方法是明智地挑选样本，然后再加入训练集；第6章将阐述一些改进的方法，同时也指出各自存在的不足。 134~135

最近邻法起源于几十年以前，统计学家在20世纪50年代早期就分析了k最近邻法。如果训练实例的数量很大，直观感觉需要使用不止一个最近邻，但是，如果实例的数量非常少，很明显这种方法是危险的。当k和实例的数量n都变成无穷大，使得$k/n \rightarrow 0$时，那么在数据集上产生的误差概率将达到理论上的最小值。早在20世纪60年代最近邻法就已经被采纳为分类方法，并且在模式识别领域已经广泛使用了30多年。

最近邻分类法的速度之慢是众所周知的，直到20世纪90年代早期开始使用kD树，尽管kD树数据结构本身发展要早得多。在实践中，当实例空间的维数增加时，这些树就变得效率很低，只有当属性数量很小时，最高为10，它才有应用价值。球树是最近才研究的，是属于一种更为通用结构的一个实例，这个通用结构有时称为测量树（metric tree）。由一些高级的算法创建的测量树能够成功地处理数千维的实例空间。

采用将所有训练实例压缩到多个区域里的方法来取代存储所有实例。在第4.1节结尾部分提到一个非常简单的技术，记录在训练数据上所观察到的每一个属性值和每一种类别上的区域。对于给出的一个测试实例，找出测试实例各个属性值所在的区域，选择与测试实例属性值区域正确对应数量达最多的那个类作为这个实例的类别。一个更为精细的技术是对每个属性建立多个区间，并且使用训练集在每个属性上对每个属性值区间统计每个类出现的次数。数值属性可以被离散成多个区间，由一个点组成的区间可以用来处理名词性属性值。然后，能够判断出一个测试实例的各个属性值分别属于哪个区间，用投票的方式对测试实例进行分类，这种方法称为投票特征区间（voting feature intervals）。这些是近似的方法，但是运行的速度很快，可以用来对大的数据集进行初始分析。

4.8 聚类

不是预测实例的类别，而是将实例分成自然的组时，就需要用聚类（clustering）技术。这些聚类想必反映了在实例所属的某个领域中的一些运作机制，这些机制导致一些实例之间彼此十分相似，而有别于其他实例。自然聚类所需要的技术不同于我们目前学到的分类和关联学习的方法。 136

正如在第3.9节所述，有多种表示聚类结果的方法。识别出的组可以是排他的，因此任何实例只能属于其中的某一个组；或者是可以重叠的组，因此一个实例可以落入几个组；或者是以概率的形式，一个实例是以一定的概率分属于每个组；或者是分等级的，在顶层将实例大致地进行分组，随后每一个组再被进一步细分，也许所有路径最终都要到达一个单独实例。对这些可能方法的选择应该由运作机制的自然属性所支配，这些运作机制被认为是特定聚类现象的依据。然而，因为这些运作机制很少被认知，毕竟聚类是真正存在的，我们正试图去发现它——再加上一些实践运用上的原因，选择通常是由现存的聚类工具所支配。

下面将考察一个算法，这个算法将在数值领域内形成聚类，把实例划分到不相交的聚类

上。正如基于实例学习的基本最近邻法一样，它是一个简单明了的技术，已经使用了几十年。在第6章将讨论一些新的聚类方法，这些方法将产生递增聚类和概率聚类。

4.8.1 基于距离的迭代聚类

经典的聚类技术称为k均值（k-means）。首先，指定所需寻找的聚类个数，这便是参数k。然后随机选出k个点作为聚类的中心。根据普通的欧几里得距离量度，将所有的实例分配到各自最靠近的聚类中心。下一步是计算出实例所在的每个聚类的质心，或者平均值，这就是“均值”部分。这些质心将成为各个聚类的新的中心值。最后，用新的聚类中心重复整个过程。迭代过程不断继续直到在连续的几轮里，每个聚类上分到的点与在上一轮分到的点相同，此时聚类的中心已经固定，并且会永远保持。

这个聚类法简单并且有效。容易证明选择质心作为聚类的中心，使得聚类中每一个点到中心的距离平方之和达到最小。一旦迭代过程的结果趋于稳定，每一个点被分配到离它最近的聚类中心，所以最终是将所有点到它们各自聚类中心的距离平方之和最小化。但是，它只是一个局部的最小值，并不能保证是一个全局的最小值。最终的聚类对初始的聚类中心相当敏感。在初始随机
137 选择上的微小变化会造成完全不同的聚类结果。实际上，通常不可能找到全局优化的聚类，这也是所有实际应用聚类技术的真实现状。为了增加找到全局最小值的机会，人们经常需要用不同的初始选择多次运行算法，然后从中选择一个最佳的结果，即距离平方之和最小的那个。

可以容易地设想出一个用k均值方法聚类失败的情况。假设，在一个二维空间上有4个实例分布在一个矩形的（4个）顶端。处于短边两端的实例分别形成两个自然的聚类。但是，如果两个初始聚类的中心落在长边的中点上，将会产生一个稳定的结构，无论长边和短边之差如何之大，所产生的两个聚类中的每一个都将拥有位于长边两端的两个实例。

4.8.2 快速距离计算

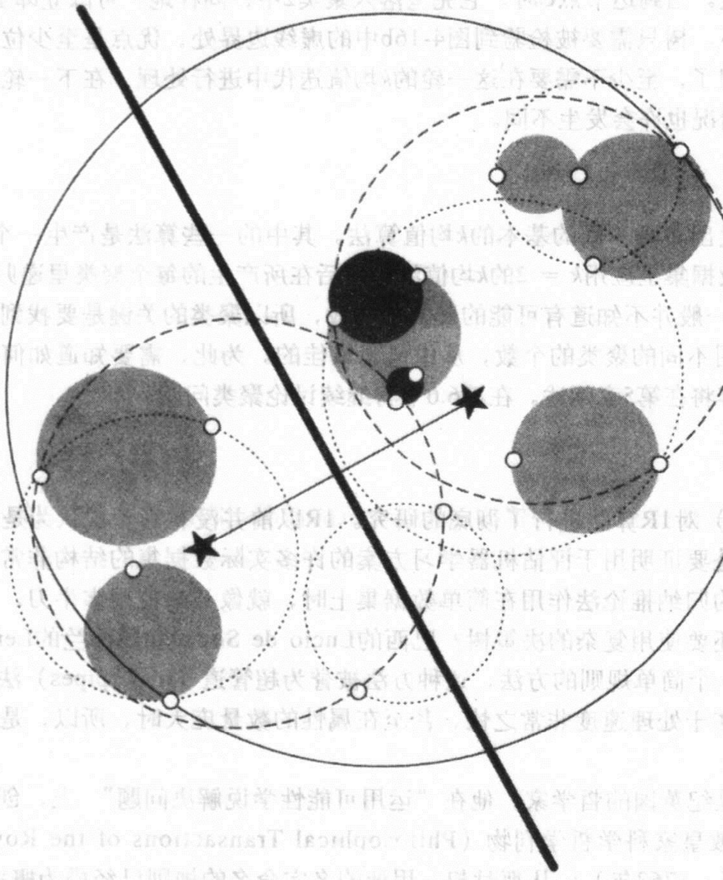
k均值聚类算法通常需要多次迭代，每次都要计算每一个实例到k个聚类中心的距离，从而决定它的聚类。利用一些简单的近似法可以使得计算速度大大提高。例如，可以将数据集投影，然后按照选定的轴进行分裂，来取代由选择最近的聚类中心所意味着使用的任意超平面的分裂法。但是所得到的聚类的质量不可避免地会降低。

这里介绍一个更好的加速法。寻找最近聚类中心和用基于实例学习方法寻找最近邻差别并不很大。那么是否同样可以借用kD树和球树这两种有效的解决方案？当然可以！实际上，可以采用一个更加有效的方法，因为在k均值的每一次迭代过程中，所有的数据点都被一起处理，而在基于实例的学习中，测试实例被个别处理。

首先，为所有的数据点，创建一个kD树或者球树，它们在整个聚类过程中将保持不变。每一次k均值的迭代过程产生一组聚类中心，所有数据点必须经检验后，分配到最近的聚类中心。一种处理数据点的方法是从树的根节点向下直到到达叶节点，然后分别检查叶节点上每个点，从而寻找它的最近聚类中心。但是，也许一个较高位置的内部节点所代表的区域会完全落入某个单独的聚类中心所涉及范围内。在这种情况下所有位于那个节点下的数据点被一次处理完。

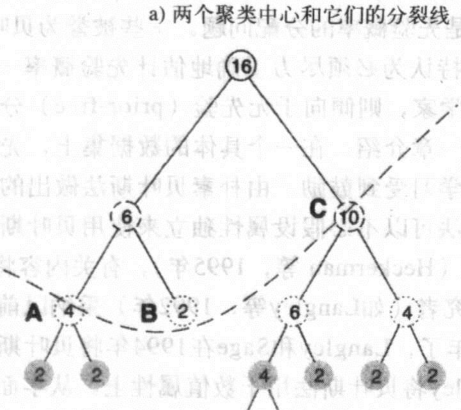
最终目的是通过计算数据点所拥有的质心，为聚类中心寻找新位置。计算质心是利用计算聚类中数据点之连续向量和，并且统计到目前为止数据点的个数。最后，用向量和除以统
138 计个数就得到质心。假如在树的每一个节点上保存该节点拥有的数据点向量之和，以及数据点的个数。当整个节点落入某个聚类范围内时，那么那个聚类计算总值便能立刻得到更新。否则，则需深入查看节点内部，递归地沿树向下处理。

图4-16使用的是与图4-14相同的实例和相同的球树，但两个聚类中心由两个黑色的星表示。因为所有的实例都被分配到最近的聚类中心，所以实例空间被一条粗直线分成两个部分，图4-16a。从图4-16b树的根节点开始，每个聚类的向量和以及每个聚类拥有的数据点的统计个数



139 | 140

a) 两个聚类中心和它们的分裂线



b) 与之相对应的树

图4-16　一个球树

都被初始化为0。处理是自上而下递归地进行的。当到达A节点时，在A上的所有数据点将落入聚类1，所以聚类1的总和数以及数据点统计数可用节点A的总和数以及数据点统计数进行更新，到此为止。然后经递归返回到B节点，因为这个球跨越了聚类的边界，所以在它上面的数据点必须分别检验。当到达节点C时，它完全落入聚类2中，同样地，可以立即更新聚类2，并不再需要继续往下。树只需要被检验到图4-16b中的虚线边界处，优点是至少位于虚线以下的节点不需要被处理了，至少不需要在这一轮的k均值迭代中进行处理。在下一轮迭代中，聚类中心将会改变，情况也许会发生不同。

4.8.3 讨论

现在已经开发出多种不同的基本的k均值算法。其中的一些算法是产生一个分级的聚类，它是通过在整个数据集上应用$k = 2$的k均值法，然后在所产生的每个聚类里递归地重复。

如何选择k？一般并不知道有可能的聚类的个数，所以聚类的关键是要找到这个数量。一种方法是试着采用不同的聚类的个数，从中选出最佳的。为此，需要知道如何评估机器学习的效果，有关内容将在第5章阐述。在第6.6节将继续讨论聚类问题。

4.9 补充读物

Holte（1993）对1R算法进行了彻底的研究。1R以前并没有真正被认为是一种机器学习的方法，其目的是要证明用于评估机器学习方案的许多实际数据集的结构非常简单，而那些被赋予了高能量的归纳推论法作用在简单数据集上时，就像杀鸡使用宰牛刀。当一个简单规则可行时为什么还要使用复杂的决策树？巴西的Lucio de Souza和新西兰的Len Trigg提出了为每一个类产生一个简单规则的方法，这种方法被誉为超管道（hyperpipes）法。一个异常简
139~140 单的算法的优势在于处理速度非常之快，甚至在属性的数量庞大时，所以，是一个十分可行的方案。

贝叶斯是18世纪英国的哲学家，他在“运用可能性学说解决问题” 上，创立了他的概率理论，发表在伦敦皇家科学哲学刊物（Philosophical Transactions of the Royal Society of London）（贝叶斯，1763年）；从那时起，用他的名字命名的规则已经成为概率理论的基石。在实际中贝叶斯规则应用的难点是先验概率的分配问题。一些被誉为贝叶斯专家的统计学家，把贝叶斯规则当作真理，并且坚持认为必须尽力正确地估计先验概率——尽管这些估计通常是主观的。而另外一些非贝叶斯学家，则倾向于无先验（prior-free）分析，产生一个统计的置信度区间，这部分内容将在下一章介绍。在一个具体的数据集上，先验概率通常相当容易估计，这使采用贝叶斯方法进行学习受到鼓励。由朴素贝叶斯法做出的属性独立的假设是一个巨大的障碍，然而，有一些方法可以不必假设属性独立来使用贝叶斯分析。结果模型称为贝叶斯网络（Bayesian networks）（Heckerman 等，1995年），有关内容将在第6.7节介绍。

贝叶斯技术在被机器学习研究者（如Langley等，1992年）采纳以前已经在模式识别领域（Duda和Hart，1973年）应用20年了，Langley和Sage在1994年将贝叶斯法用于存在冗余属性的数据集上，1995年John和Langley将贝叶斯法用于数值属性上。从字面上看，朴素贝叶斯是一个简单的方法，但是在某些场合，一点也不简单。1998年McCallum和Nigam研究出专门针对文本分类的多项朴素贝叶斯模型。

Quinlan（1986年）发表了经典的决策树归纳论文，正是他描述了基本的ID3产生过程，

在本章中对这个算法进行了开发。在Quinlan（1993年）的一本经典书中给出了一个这个方法的综合描述，包括C4.5系统的改进，并且列出了用C语言开发的完整的C4.5系统。Cendrowska（1987年）开发了PRISM，也是他推出了隐形眼镜数据集。

关联规则的提出和讨论出现在数据库文献而不是机器学习文献中。对关联规则的研究着重于如何处理数量庞大的数据，而不是在有限数据集上对算法的测试和评估方法的研究。本章介绍的Apriori算法是由Agrawal和他的同事（Agrawal等1993年，1993年；Agrawal和Srikant，1994年）共同开发的。Chen等（1996年）出版了一个关联规则数据挖掘的调查报告。

在很多标准的统计书中描述了线性回归法，Lawson和Hanson（1995年）介绍了一个特别成熟的处理方法。在20世纪60年代，使用线性模型进行分类引起了人们的极大关注，Nilsson（1965年）提供了一个极好的参考书目，他将线性边界阈值单位（linear threshold unit）定义 141
为判定一个线性函数的结果是否大于或者小于0的一个二元测试，并将线性机器（linear machine）定义为一系列的线性函数，每一个类对应一个线性函数，将线性函数在一个未知样本上所得到的值进行比较，其中最大值所对应的类就是这个样本的预测类别。很久以前，一本有影响的书（Minsky和Papert，1969年）中声明感知器存在基本原理上的局限，所以没有受到重视。然而一些更复杂的线性函数系统以神经网络的形式在近几年得到重新发展，第6.3节将具体讨论神经网络。1989年Nick Littlestone在他的博士论文中介绍了Winnow算法（Littlestone，1988年，1989年）。最近多反馈的线性分类法已经在一个称为堆栈（stacking）的操作上找到新的应用领域，它结合了其他机器学习算法的结论，有关内容将在第7章介绍（看Wolpert，1992年）。Friedman（1996年）讨论了成对分类技术；Fürnkranz（2002年）对此展开了进一步研究，Hastie和Tibshirani（1998年）将分类技术扩展，使用成对合并预测概率。

Fix和Hodges（1951年）首先对最近邻方案进行分析，Johns（1961年）开创了最近邻法在分类问题方面的使用。Cover和Hart（1967年）给出了经典的理论结论：对于足够大的数据集，它产生的误差概率不会超出理论最小值的两倍；Devroye等（1996年）指出k最近邻法是当增大k和n并存在$k/n \to 0$时，将逐渐趋于最佳。经过Aha（1992年）的研究努力最近邻法在机器学习领域受到重视，Aha指出基于实例的学习法经过结合干扰样本修剪和属性加权法以后，与其他机器学习方法相比，性能更加优越。我们将在第6章中介绍。

kD树数据结构是由Friedman等（1977年）开发。我们的描述紧紧遵循了Andrew Moore在他博士论文里的描述，Andrew和Omohundro（1987年）一起拓展了kD树在机器学习领域的使用。Moore（2000年）讨论了一些成熟的创建球树的方法，这些方法在拥有数千个属性的数据集上表现优异。本书使用的球树例子是从卡内基－梅隆大学Alexander Gray的教学笔记上摘取的。在第4.7节最后部分讨论小节中提到的投票特征区间是由Demiroz和Guvenir（1997年）描述的。

k均值算法是一个经典的技术，有很多有关的描述和版本（如Hartigan，1975年）。我们所选用的使用球树取代kD树来加速k均值聚类法，是由Moore和Pelleg（2000年）在他们的X均值聚类算法中开创的。这种算法包含的其他创新将在第6.6节介绍。 142

第5章 可信度：评估机器学习结果

评估是数据挖掘能否取得真正进展的关键一环。我们已经见到了许多自原始数据中推出某种结构的方法。在下一章中还将介绍更为细致的新的方法。采取何种推论方法来解决某一具体问题，需要对不同的推论方法进行系统的比较评估。评估并不像看上去那样简单。

问题在哪里？我们有训练集数据，当然可以观察用不同推论方法在这个训练集上所得的不同结果。然而，我们很快会发现，在训练集上表现好的绝不意味着在独立的测试集上会有好的表现。我们需要能预测在实践中性能表现的评估方法，这个预测基于所能得到的任何数据上的实验。

当数据来源很充足时，这并不是问题。只要在一个大的训练集上建模，然后在另外一个大的测试集上验证。虽然数据挖掘时常牵涉到一些大型的数据库，特别是在市场、销售和客户支持应用当中，但是也经常出现数据（有质量的数据）匮乏的情形。比如在第1章（第1.3
143 节）中提到的海面浮油应用，训练数据必须经过人工探测和标记方可使用，这是一个非常专业，且劳动力密集的过程。甚至在信用卡申请（第1.3节）实例中，只有一千个适当的训练实例。在供电量数据库(第1.3节)中，如追溯到15年前，共有5000天，但其中只有15个圣诞节、15个感恩节、4个2月29日和4个总统大选日。在电子机械诊断应用(第1.3节)中，虽有20年的使用记录，但是其中只包含了300个可用的故障例子。虽然市场和销售应用(第1.3节)肯定涉及到大型数据库，但是许多其他应用时常依赖于一些专家的专业特长，以至于数据缺乏。

基于有限数据的预测性能评估是一个有趣的问题，仍存争议。预测性能评估有许多不同的技术，其中重复交叉验证（repeated cross-validation）在实践中或许是适合大部分有限数据情形的评估方法。比较不同的机器学习方法运用在某个给定问题上也并非易事，需要用统计学测试来确定那些明显的差异并非是偶然发生的。到目前为止，我们默认所要预测的是对测试实例进行正确分类的能力。然而，在某些情况下，却要涉及到预测分类概率而非类别本身，另外一些情况需要预测数值而不是名词性属性值。需要视不同情形而使用不同的方法。接下来我们要看一下成本问题。在大多数的实际数据挖掘情形中，错误分类误差的成本是由误差类型所决定的，如错误是将一个肯定的例子错误地归类为否定的，还是反过来（将否定的例子归类为肯定的——译者注）。在做数据挖掘及性能评估时，这些成本的考虑是非常重要的，所幸的是采用一些简单的技术能使大多数的学习方案具有成本敏感性，而不必探究算法本身。最后，从整体上看，评估有着迷人的哲学含义，哲学家们已对如何评估科学理论辩论了2000年，此议题亦是数据挖掘的一个焦点，因为从本质上来看，我们挖掘的是数据“理论”。

5.1 训练和测试

对于分类问题，自然是采用误差率（error rate）来衡量一个分类器的性能。分类器对每个实例进行类预测，如果预测正确，则分类成功，反之则分类错误。误差率就是所有错误在整个实例集中所占的比率。误差率是对分类器总体性能的一个衡量。

144 当然，我们感兴趣的是分类器对未来新数据的分类效果，而非旧数据。训练集中每个实

例的类都是已知的，正因为如此才能用它进行训练。通常我们不是对学习这些实例的分类感兴趣，除非是要进行数据整理而非预测。问题是在旧数据集上得出的误差率是否可以代表在新数据集上的误差率？答案是否定的，如果分类器是用旧数据集训练出来的。

这是一个令人惊讶、亦是非常重要的事实。分类器对训练集进行分类而得出的误差率并不能很好反映分类器未来的工作性能。为什么？因为分类器正是通过学习这些相同的训练数据而来的，因此该分类器在此训练数据集上进行的任何性能评估结果都是乐观的，而且是绝对乐观。

我们曾在劳工关系数据集中见过这样的例子。图1-3b是由训练数据直接产生的，图1-3a则是经过修剪处理的。若用训练数据对二者进行评估，前者似乎更准确。但若用独立的测试数据对二者进行评估，前者的表现很可能会不如后者，因前者与训练数据过度拟合。根据在训练数据上得出的误差率，前一个决策树比后一个决策树要好，但这并不能反映它们将来在独立的测试数据上的表现。

用训练数据进行测试所产生的误差率称为重新代入误差（resubstitution error），因为它是将训练实例重新代入由这些训练实例而产生的分类器进行计算的。虽然它不能可靠地反映分类器在新数据上真实的误差率，但依然是有参考价值的。

为了能预测一个分类器在新数据上的性能表现，需要一组没有参与分类器建立的数据集，并在此数据集上评估分类器的误差率。这组独立的数据集叫测试集(test set)。必须假设训练数据和测试数据都是某特定问题的代表性样本。

在某些情况下，测试数据也许和训练数据存在着明显差别。例如，在1.3节中提到的信用风险问题。假设银行现有来自纽约和佛罗里达州两个分行的训练数据，银行想用其中的一个数据集来训练出一个分类器，然后看看此分类器用在内布拉斯加州新分行结果会如何。可以将佛罗里达州的数据作为测试数据，来评估由纽约数据训练的分类器；同时将纽约的数据作为测试数据，来评估由佛罗里达州数据训练的分类器。如果在训练前就将两个分行的数据合并，在测试数据上的性能评估恐怕不能较好地反映此分类器将来用于另一个完全不同的州的分类性能。

测试数据无论如何不能参与分类器的创建，这点非常重要。举例来说，有些学习方案包括两个阶段，第一阶段是建立基本结构，第二阶段是对结构所包含的参数进行优化，这两个阶段需要使用不同的数据集。或者，先用训练数据尝试多种方案，然后用新的数据集对这些分类器进行评估，找出最好的。但是所有这些数据都不可用于估计未来的误差率。这就是人们经常提到的三种数据集：训练数据、验证数据（validation data）和测试数据。训练数据用在一种或多种学习算法中创建分类器；验证数据用于优化分类器的参数，或用于选择（参数）；测试数据则用于分类器最终经过优化的方法的误差率计算。三个数据集必须保持独立性，验证数据集必须有别于训练数据集以获得较好的优化或选择阶段的性能，同时测试数据集也必须有别于其他两个数据集以获得对真实误差率的可靠估计。 145

一旦误差率已定，可以将测试数据合并到训练数据中，由此产生新的分类器应用于实践中。这并没有错，使用尽可能多的数据来建立分类器，这是一种实践中常用的方法。重要的是误差率不能源自于这些数据。同样，一旦验证数据已被使用（也许是用于选择决定最好的学习方案），那么可以将验证数据合并到训练数据中，使用尽可能多的数据重新训练学习方案。

如果数据源充足，一切都没有问题。可以取一个大的样本用来训练，取另一个不同、且

独立的大样本数据用于测试。假设这两个样本都具有代表性，那么由测试样本得出的误差率将反映其未来的真实性能。一般来说，训练样本越大，所建的分类器性能越好。虽然当训练样本超过一定的限度，性能提高将会有所减缓。测试样本越大，误差估计越准确。误差估计的准确性可从统计学角度进行量化，这点我们将在下一节中分析。

问题在于当数据源不充足时。在许多情况下，训练数据必须经过人工操作分类获得，测试数据亦是如此，使估计出错。这使可用于训练、验证和测试的数据量非常有限。怎样才能最好地利用这样一个有限的数据集？将这个数据集当中的一部分数据置于一旁用于测试称为旁置法（holdout method），剩余的数据用于训练（如有必要可再留部分用于验证）。这里出现一个难题：要得到一个好的分类器需要尽可能多的数据用于训练；而要得到一个准确的误差估计，也需要尽可能多的数据用于测试。我们将在5.3节和5.4节回顾现有的解决方法来解决这个难题。

5.2 预测性能

假设用测试集进行分类器的误差测试，得到一个误差率，为25%。在这一节中实际上要
146 谈论的是成功率而不是误差率，那么对应的成功率是75%。这仅是一个估计值，那目标总体真正的成功率是多少呢？当然，预计应接近75%。但到底有多接近？5%以内？10%以内？这个答案依赖于测试数据集的规模。一般来说，这75%的成功率若是基于10 000个实例的测试集而得到的，它的可信度要高于基于100个实例而得到的。但到底高多少？

为了回答这个问题，需要一些统计学推理。在统计学中，一连串不是成功便是失败的独立事件，称为伯努利程式（Bernoulli process）。抛掷硬币是一个经典实例。每次抛掷是一个独立事件。假使总是预测正面出现，除了用“正面”或“反面”来描述，每次抛掷结果亦可视为“成功”或“失败”。假设硬币正、反面是有偏差的，但并不知道正面的几率有多大。如果抛掷100次，其中75次是正面，便拥有了一个与上述在测试集上达75% 成功率的分类器极为相似的情况。应该怎样描述真正的成功率呢？换句话说，想像存在一个伯努利程式，一个有偏差的硬币，它的真实成功率为p（但是未知的）。在N次测试中，S次是成功的：这样观察到的成功率便是$f=S/N$。问题是，这对于了解真实的成功率p有何帮助？

这个问题的答案通常被表达为一个置信度区间（confidence interval），即真实成功率p以某个特定的置信度存在于某个特定的区间中。例如，如果观察到N=1 000次测试中，存在S=750次是成功的，这表明真实成功率在75%左右。但到底有多接近75%？若置信度为80%，真实成功率p则在73.2%和76.7%之间。如果观察到N=100次测试中，存在S=75 次是成功的，这亦表明真实成功率在75%左右。但是由于试验次数较少，同样是80%的置信度，真实成功率p的区间则较宽，伸展为69.1%和80.1%之间。

这很容易定性，但如何来给它们定量呢？推理如下，真实成功率为p的单次伯努利测试的平均值（mean）和方差(variance)分别为p和$p(1-p)$。如果一个伯努利程式中含有N次测试，那么期望成功率$f=S/N$是一个平均值为p的随机变量，而方差则随N递减为$p(1-p)/N$。N值较大时，这个随机变量的分布接近于正态分布。这些都是统计学的知识，这里将不作推导。

一个随机变量X，平均值为0，落入某个置信范围宽度为$2z$的概率为

$$\Pr[-z < X < z] = c$$

147 对于正态分布，大多数统计学课本的背后都有c值和对应的z值的列表。但是这种传统列

表形式(与此)略有不同：它们给出的是随机变量X将落在某范围之外的置信度，而且只给出上界值：

$$\Pr[X \geqslant z]$$

这称为单边（one-tailed）概率，因它只涉及整个分布的上界端。正态分布是对称的，因此它的下界端的概率

$$\Pr[X \leqslant -z]$$

是一样的。

表5-1给出一个示例。同其他的正态分布表一样，这里也假设随机变量X的平均值为0，方差为1。或者，也可以说z是距离平均值（有多少个）标准差的衡量。因此，$\Pr[X \geqslant z]=5\%$意味着X落在高于平均值1.65个标准差以上的几率是5%。由于分布是对称的，X落在距离平均值1.65个标准差以外（高于或低于）的几率是10%，或者说

$$\Pr[-1.65 < X < 1.65]=90\%$$

现在所要做的就是减小随机变量f，使它的平均值为0，并将方差单位化。为此我们减去平均值p并除以标准差$\sqrt{p(1-p)/N}$。从而得到

$$\Pr\left[-z<\frac{f-p}{\sqrt{p(1-p)/N}}<z\right]=c$$

表5-1 正态分布的置信边界

$\Pr[X \geqslant z]$	z
0.1%	3.09
0.5%	2.58
1%	2.33
5%	1.65
10%	1.28
20%	0.84
40%	0.25

以下是得到置信边界(confidence limits)的过程。对于某一给定的置信度值c，在表5-1中查到相应的z值。在查表之前先用1减去c，再将结果除以2。如此，$c = 90\%$ 则要用5% 在表中查找。中间置信度可用线性内插法。然后将上面的不等式写成等式，再将其转换为p的表达式。 148

最后一步涉及到解二次方程。虽然不难解，但得出的却是一个看似恐怖的置信边界表达式：

$$p=\left(f+\frac{z^2}{2N}\pm z\sqrt{\frac{f}{N}-\frac{f^2}{N}+\frac{z^2}{4N^2}}\right)\Bigg/\left(1+\frac{z^2}{N}\right)$$

表达式中的 ± 符号给出两个p值，分别代表置信上界和下界值。虽然公式看起来有点复杂，但在实际应用中并不太难。

用这个公式可得到前面例子中提到的真实成功率数值。设定$f=75\%$，$N = 1000$，$c = 80\%$（因此$z = 1.28$），得到p值区间[0.732, 0.767]。当$N = 100$，在相同的置信度下，可得到p值区间[0.691, 0.801]。要注意的是，正态分布的假设只有在N值较大时（比如$N>100$）才有效。因此，当$f=75\%$，$N = 10$时得到置信边界为[0.549, 0.881]，这个结论是应有所保留的。

5.3 交叉验证

现在来考虑当训练和测试数据数量有限时该如何处理。旁置法保留一定数量的数据作为测试用，剩余的数据用于训练（如有需要可再留一部分数据用于验证）。在实践中，一般保留三分之一的数据用于测试，而剩余的三分之二用于训练。

当然，也许不巧：用于训练（或测试）的样本不具代表性。一般，你无法说一个样本具有或不具有代表性，但有个简单的检测方法值得一试。每个类在整个数据集中所占的比例在训练集和测试集中也应体现出相应的比例。如果某一类的样本不巧在训练集中一个也没有，很难想象由这样的训练集训练出来的分类器将来对属于这一类的样本数据进行分类时会有好的表现。更糟糕的是，由于训练集中没有属于这一类的实例，从而使得这个类不可避免地在测试集中过多地出现！因此，在随机取样时必须确保在训练集和测试集中每个类各自应有的比例。这个过程叫分层（stratification），我们还会提到分层旁置（stratified holdout）。虽然很有必要进行分层，但分层只能为防范训练集和测试集数据的样本代表性不一致提供一个基本的安全措施。

一个更为通用的方法来减少由于旁置法取样而引起的任何偏差，就是重复整个过程。用
149 不同的随机样本重复进行几次训练和测试。每次迭代过程中，随机抽取一个特定的比例（比如三分之二）的数据进行训练，也可能经过分层处理，剩余的数据用于测试。将每次不同迭代过程中所得的误差率进行平均得到一个综合误差率。这就是重复旁置法（repeated holdout method）的误差率估计。

你也许会考虑在单次旁置的过程中，交换训练集和测试集的角色，即用测试集数据进行训练并用训练集数据进行测试，然后将两次不同结果平均一下，以此来减少训练集和测试集数据代表性不一致所产生的影响。不幸的是，只有当训练集和测试集数量比例为50:50时才似乎真正合理，可这个比例通常不太理想，还是要用半数以上的数据进行训练效果比较好，即使是耗费了测试数据。然而，一个简单的变异方法造就了一个重要的统计学技术，即所谓的交叉验证（cross-validation）。在交叉验证中，先要决定一个固定的折数（number of folds），或者说是数据分区数。假设定为三折，那么数据将被大致均分成三个分区，每个分区被轮流用于测试而剩余的则被用于训练。也就是说，用三分之二的数据进行训练，三分之一的数据进行测试，重复此过程三次，从而每个实例恰好有一次是用于测试的。这就是所谓的三折交叉验证，若同时采用了分层技术（经常如此），这就是分层三折交叉验证（stratified threefold cross-validation）。

给定一个数据样本，预测某种机器学习技术误差率的标准方法就是使用分层10折交叉验证。数据被随机分割成10个部分，每一个部分中的类比例与整个数据集中的类比例要基本一致。每个部分依次轮流被旁置，其余十分之九的数据则参与某一个学习方案的训练，而旁置的数据集则用于计算误差率。这样，学习过程共进行10次，每次使用不同的训练集（含有许多相同数据）。最后，将10个误差率估计值平均而得出一个综合误差估计。

为什么是10次？经过大量的试验，使用大量的数据集，采用不同的学习技术，表明10折正是获得最好的误差估计的恰当选择，而且也有一些理论根据可以证明这一点。虽然这个论点还不是最后的论断，在机器学习和数据挖掘领域有关什么才是最好的评估方案的问题至今还持续着激烈的争辩，但是10折交叉验证法在实践中被认为是标准方法。试验还证明采用分层技术能使结果稍有改进，因此当数据集数量有限时，分层10折交叉验证法被当作标准评估

技术。值得注意的是，无论是在分层或是进行10折分割时都不必很严格，只要能将数据集分割成大致相等的10个部分，每部分中各个类的比例基本恰当就可以了。统计评估不是一门精确的科学。另外，10折也并非是有魔力的，5折或20折交叉验证似乎也相差无几。 150

为了得到可靠的误差估计，单次的10折交叉验证恐怕还不够。采用相同的学习方案，在相同的数据集上进行不同的10折交叉验证，常常会得到不同的结果，这是由于在选择确定折本身时受到随机变化的影响。分层技术可减少变化，但不能完全消除随机变化。当需要一个准确的误差估计时，标准的程序是重复10次交叉验证——即10次10折交叉验证，然后取其平均值。这将使原始数据中十分之九的数据被代入学习算法中100次。可见，获得好的测试结果是一项计算密集型的任务。

5.4 其他估计法

10折交叉验证是衡量一种学习方案使用在某一数据集上的误差率的标准方法，为得到可靠的结果，建议使用10次10折交叉验证。除此之外，还有许多其他方法可行，其中两个特别普及的方法就是留一（leave-one-out）交叉验证和自引导法（bootstrap）。

5.4.1 留一法

留一（leave-one-out）交叉验证其实是n折交叉验证，其中n是数据集所含实例的个数。每个实例依次被保留在外，而剩余的所有实例则用于学习方案的训练。它的评估就是看对那个保留在外的实例分类的正确性，1或0分别代表正确或错误。所有n个评估结果（数据集中的每个成员各产生一个结果）被平均，得到的平均值便是最终的误差估计。

这个方法之魅力所在有两个方面。第一，每次都使用尽可能多的数据参与训练，从而增加了获得正确分类器的机会。第二，这个方法具有确定性：无需随机取样。没有必要重复10次或任何重复操作，因为每次的结果都将是一样的。然而，它的计算成本也是相当高的，整个学习过程必须执行n次，这对一些大的数据集来说通常是不可行的。不过，留一法似乎是提供了一个机会，即最大限度地从一个小的数据集里获得尽可能正确的估计。

但是除了计算成本高之外，留一交叉验证法还有一个缺点。它不但不能进行分层（stratified），更糟的是它一定是无层样本。分层是使测试集数据拥有恰当的类比例，当测试集中只含一个实例时，分层是不可能实现的。举个例子，虽然极不现实，但却非常戏剧性地描述了由此而引起的问题。想象有一个完全随机的数据集，含有数量相等的两个类。面对一个 151
随机数据，所能给出的最好预测便是预测它属于多数类（majority class），其真实的误差率是50%。但在留一法的每一个折里，与测试实例相反的类（在训练集上）是多数类，因此每次预测总是错的，从而导致估计误差率达100%！

5.4.2 自引导法

第二种要描述的估计方法是自引导法（bootstrap），它是基于统计学的放回抽样（sampling with replacement）程序。先前，一个样本一旦从数据集中被取出放入训练集或测试集，它就不再被放回。也就是说，一个实例一旦被选择一次，就不能再次被选择。这就像踢足球组队，不能选同一个人两次。但是数据集实例不是人，大多数的学习方案还是可以使用相同实例两次的，并且在训练集中出现两次会产生不同的学习结果。（数学行家将会注意到，

要是同一个对象可以出现一次以上，我们其实不是在谈论“集”。）

自引导法的想法是采取放回抽样数据集的方法来形成训练集。我们将阐述一个特例，它非常神奇（原因很快会显现）称为0.632 自引导。一个拥有n个实例的数据集进行了n次的放回抽样，从而形成了另一个拥有n个实例的数据集。因为在第二个数据集中会（几乎肯定会）有一些重复实例，那么在原始的数据集中必有部分实例未被抽样，我们将用这些实例作为测试实例。

某个具体实例不被抽样到训练集中的几率是多少呢？每次被抽取的概率是$1/n$ ，所以不被抽取的概率是$1-1/n$ 。根据抽取次数（n次），将这些概率相乘，得到

$$\left(1-\frac{1}{n}\right)^n \approx \mathrm{e}^{-1}=0.368$$

（其中e是自然对数的基数2.7183 ,不是误差率！）。这给出了某个具体实例不被抽取到的几率。因此，对一个相当大的数据集来说，测试集将包含约36.8% 的实例，而训练集将包含约63.2% 的实例（现在你该知道为什么叫0.632 自引导了吧）。训练集中包含一部分重复实例，总容量是n , 和原始数据集一般大。

用此训练集训练一个学习系统，并用测试集计算误差率，得到的将是一个对真实误差率较为悲观的估计值。这是因为虽然训练集的容量是n，但它只包含了63% 的实例，这和含有
152 90%实例的10折交叉验证法相比，不是很理想。为了补偿这点，将测试集误差率和用训练集数据计算的重新代入（resubstitution）误差率组合在一起。如前所述，重新代人误差率是对真实误差率的一个过于乐观的估计值，当然不能单独使用。自引导法将它和测试误差率组合在一起，给出最终误差率估计值e如下：

$$e=0.632\times e_{\text{测试实例}}+0.368\times e_{\text{训练实例}}$$

然后，将整个自引导过程重复进行几次，以取得不同的放回取样训练集，测试结果取平均值。

自引导方法对非常小的数据集来说，也许是最佳的误差率估计法。但是，就像留一交叉验证法一样，自引导法也有缺点，可以通过考虑一个非常特殊、人为假设的情况来描述此问题。实际上，我们考虑的数据集是一个包含两个类而且是完全随机的数据集。对任何预测规则来说，它的真实误差率都是50% 。但一个能记住训练集的方案，会给出完美的100%的重新代入评分，以致$e_{\text{训练实例}}=0$, 0.632 自引导法再将它与0.368权相乘，得出综合误差率只有31.6%（$0.632\times50\%+0.368\times0\%$），这个结论未免过于乐观了。

5.5　数据挖掘方案比较

经常需要对解决同一问题的两种不同学习方案进行比较，看哪一种更适合使用。这看来似乎很简单，用交叉验证法（或其他合适的预计方法），也许重复几次，然后选择估计误差率较小的方案。在许多实际应用中，这已是相当足够了，如果某方案在一具体数据集上的估计误差率比另一方案低，最好采用前者的模型。但是，有时差别只是源于估计错误，在某些情况下重要的是，我们要确定一个学习方案对某个具体问题的解决是否真的优于另一个方案。这是对机器学习研究者的一个挑战。如果要提出一个新的学习算法，它的提议者必须证明新算法对问题的解决是有改进的，而且要证明观察到的改进不只是估计过程中所产生的偶然结果。

这是一项给出置信边界的统计试验工作，正如我们先前谈到的，根据已给出的测试集误差率来预测其真实性能。假设有无限的数据来源，则可以用大量的数据进行训练，然后再用另一个大型的独立测试数据集进行性能评估，像先前一样得到置信边界。但是，如果差别很显著，必须确定其原由不是因碰巧使用某个具体数据集进行测试而引起的。我们要决定一个方案从整体平均上来看比另一个好还是差，这是要涵盖能从这个领域得到的所有训练集和测试集数据。训练集数据的数量当然会影响性能，因此所有的数据集大小要一致：实际上，也可用不同大小的数据集进行重复试验以得到一条学习曲线。 153

此刻，假定数据来源是无限的。为明确起见，假设用交叉验证法来进行误差估计（其他估计法，如重复交叉验证，也是同样可行的）。可以提取几个一样大小的数据集，对每个学习方案、每个数据集用交叉验证法得到一个正确率估计，然后计算出正确率估计的平均值。每个交叉验证试验产生一个不同的、独立的误差估计，而我们感兴趣的是这个平均正确率要涵盖所有可能的相同规模数据集，以及这个平均值是否在使用某一方案时较大一些。

从这个角度看，我们正在试图判断一组样本的平均值，是否明显地高于或低于另一组样本平均值，这里的样本是指对于从某个领域抽取的不同数据集进行交叉验证所得到的估计值。这是一项统计学工作，称为t测试（t-test）或学生氏t测试。由于同样的交叉验证试验可以用于两个学习方案，使在每个数据集上的试验都能获得配对的结果，因此可以使用t测试的这种更为敏感的形式，称为成对的t测试（paired t-test）。

现在需要定义一些符号。有一组样本$x_1, x_2, \cdots, x_k$，是使用某种学习方案进行连续的10折交叉验证而得到的。第二组样本$y_1, y_2, \cdots, y_k$，是使用另一种学习方案进行连续的10折交叉验证而得到的。每个交叉验证估计是使用不同的数据集（但所有的数据集大小相同，且来源于同一领域）而产生的。如果完全相同的交叉验证分区被用于两个学习方案，将会得到最好的结果。因此，x_1和y_1是在使用相同的交叉验证分割条件下而得到的，x_2和y_2也是如此，依此类推。第一组样本的平均值用$\bar{x}$来表示，第二组用$\bar{y}$来表示。我们要试图判定$\bar{x}$和$\bar{y}$是否有显著的差别。

如果样本足够，独立样本（$x_1, x_2, \cdots, x_k$）的平均值（$\bar{x}$）应呈正态（也是高斯）分布，尽管这个分布是潜在于样本本身的。称真实的平均值为μ。如果知道正态分布的方差，那么可将其平均值降为0并将方差单位化，给出样本平均值（$\bar{x}$）就能得到μ的置信边界。可是，方差是未知的，唯一的办法就是从样本组中将其估计出来。 154

这并不难，$\bar{x}$的方差估计可以将由样本$x_1, x_2, \cdots, x_k$计算而来的方差（称σ_x^2）除以k而得到。但是我们不得不估计方差这个事实，多少使结果有点变化。可用下列公式将$\bar{x}$的分布的平均值降为0并将方差单位化

$$\frac{\bar{x}-\mu}{\sqrt{\sigma_x^2/k}}$$

因为方差只是估计值，这不能算是正态分布（虽然当k值足够大时呈现出正态分布）。然而，它是一种称为$k-1$自由度的学生氏分布。在实践中这意味着要使用学生氏分布的置信区间表，来代替原先的正态分布置信区间表。表5-2列出了自由度为9度时（这是使用10折交叉验证法平均所应当用的正确度数）的置信边界。如果将它与表5-1比较，你会发现学生氏分布稍许保守一点。对于一个给定置信度，学生氏分布区间稍宽，这正反映了不得不进行方差估计所带来的不确定性因素。不同的自由度有不同的列表。如果自由度超过100度，学生氏分布

与正态分布的置信边界非常接近。和表5-1一样，表5-2中的数值亦是“单边”置信区间。

要判断平均值 $\bar{x}$ 和 $\bar{y}$（都是k个样本的平均值）是否相等，我们来考虑每组对应的观察点之间的差值d_i，$d_i = x_i - y_i$。这样考虑是合理的，因为这些观察点都是成对的。这些差值的平均值正是两个平均值（$\bar{x}$ 和 $\bar{y}$）之间的差，$\bar{d} = \bar{x} - \bar{y}$。和平均值本身一样，平均值差值也是$k-1$自由度的学生氏分布。如果平均值相等，则差值为0（称为零假设（null hypothesis））；如果有显著差异，则差值将和0有着显著差距。因此，对一个给定的置信度，我们要检查其真实差值是否超过置信限度。

表5-2 9自由度的学生氏分布置信边界

$\Pr[X \geq z]$	z
0.1%	4.30
0.5%	3.25
1%	2.82
5%	1.83
10%	1.38
20%	0.88

155

首先要减小差值，使其成为平均值为0、方差单位化的变量，被称为t统计量(t-statistic):

$$t = \frac{\bar{d}}{\sqrt{\sigma_d^2 / k}}$$

这里 σ_d^2 是差值样本的方差。然后决定置信度，在实践中通常使用1%或5%。如果k是10，置信边界z就可从表5-2中得到；若不是10，应采用与k对应的学生氏分布置信边界表。用双边（two-tailed）测试比较适当，因为我们不知道x的平均值是否可能大于y的平均值，或是相反。因此，对于1%的测试，我们采用表5-2中0.5%所对应的值。如果根据上面公式得到的t值大于z或小于$-z$，便排除平均值相等的零假设，而认为这两个学习方法针对这个领域、这样的数据集容量，确实存在明显差别。

关于这个方法，有两个观察评论值得一提。第一个是技术上的：如果观察点不是配对的怎么办？换句话说，如果由于某种原因，无法让每个学习方案在同样的数据集上进行误差评估怎么办？如果每个方案使用的数据集个数不同怎么办？这些情况发生在当某人已经对一个方案进行了评估，并公布了针对某个具体领域、某个具体数据集容量的几种不同的估计（或者只是它们的平均值和方差），而我们想要将其与其他不同的方案进行比较。这时就有必要使用常规的不配对的t测试。如果如我们所假设的，平均值是正态分布的，那么平均值的差值也是正态分布的。我们计算平均值的差值 $\bar{x}-\bar{y}$，来代替计算差值的平均值 $\bar{d}$。当然这是一码事，差值的平均值等于平均值的差值，但是差值 $\bar{d}$ 的方差却不同。如果样本x_1, x_2，…，x_k 的方差是σ_x^2，样本y_1, y_2，…，y_l 的方差是σ_y^2，最好的平均值差值的方差估计是

$$\frac{\sigma_x^2}{k} + \frac{\sigma_y^2}{l}$$

就是这个方差（更准确地说是它的平方根）应当作为上述t 统计量计算公式中的分母。查询学生氏分布置信边界表所必需的自由度应保守地选择两个样本中最小的一个。从本质上看，知道观察点是成对的，便能使用更好的方法估计方差，产生一个更为严格的置信边界。

第二点评论涉及我们所做的假设，即数据来源是无限的，可以使用几个恰当容量的独立数据集。而在实践中，经常只有一个容量有限的数据集可用。该怎么办？可将整个数据集分割成多个（也许10个）子集，对每个子集分别进行交叉验证。但是，验证的综合结果只能说明一个学习方案是否适合于这个具体容量的数据集，也许是原始数据集的十分之一大小。或者，重复利用原始数据集，比如在每次交叉验证时对数据集进行不同的随机化。[⊖]然而，这样得出的交叉验证估计不是独立的，因为它们不是从独立的数据集上得来的。在实践中，这意味着，被判定为有明显差异的情形实际却未必。实际上，增加样本的数目k（即交叉验证的次数），最终将导致产生明显差异，因为t统计量值在毫无限制地增加着。 156

围绕这个问题，已经提出了多种对于标准t测试的修改建议都只是直观推断，缺乏理论依据。其中有一种方案在实践应用中似乎不错，称为纠正重复取样t测试（corrected resampled t-test）。假设使用重复旁置法来代替交叉验证法，将数据集进行不同的随机分割k次，获得对两种不同学习方案的正确率估计。每次用n_1个实例训练，用n_2个实例测试，差值d_i则根据在测试数据上的性能计算而来。纠正重复取样t测试使用经修改的统计量

$$t=\frac{\bar{d}}{\sqrt{\left(\frac{1}{k}+\frac{n_2}{n_1}\right)\sigma_d^2}}$$

和标准的t统计量计算方式几乎一致。再仔细看一下这个公式，现在t值不再容易随着k值的增加而增长了。在重复交叉验证中也可以使用这样的修改统计量，其实这只是重复旁置法的一个特例，每个交叉验证所使用的测试集是不交迭的。对于重复10次的10折交叉验证，$k=100$，$n_2/n_1=0.1/0.9$，σ_d^2则基于100个差值。

5.6 预测概率

这一节的目的就是要获得最大预测成功率。如果对测试实例的预测值与实际值一致，预测结果视为正确；如果不一致，则视为不正确。不是黑色就是白色，不存在灰色、正确，或者不正确。在许多情况下，这是最妥当的观点。如果一个学习方案被真正采用，导致一个不是正确就是错误的预测，成功与否便是一个合适的量度。有时称为0－1损失函数（0-1 loss function），如果预测正确，“损失”为0，不正确即为1。损失是一个习惯用语，虽然用受益（profit）作为术语是比较乐观的用词。 157

其他情况属于模糊边界。大多数的学习方案能将预测和概率结合在一起（如朴素贝叶斯方案）。在判定正确性时，考虑它的概率是很自然的。例如，一个概率为99%的正确预测多半会比一个概率为51%的正确预测分量更重一些；如在一个二类情况下，51%的正确预测并不比概率为51%的不正确预测好多少。是否要考虑预测的概率，要看具体应用。如果最终的应用只是要一个结果预测，并且对预测值可能性的现实评估也不会有任何奖赏，使用概率似乎不妥。如果预测值还要进一步处理，也许还要牵涉到人为的评估或是成本分析，甚至或许要作为第二阶段学习过程的输入，那么进行概率考虑也许是很合适的。

⊖ 这个方法早在本书第1版中已提倡。

5.6.1 二次损失函数

假设一个单独的实例有k种可能的结果，或者说k种可能的类。对某个给定的实例，学习方案对这些类提出一组概率向量$p_1, p_2, \cdots, p_k$（这些概率的和为1）。这个实例的真实结果是这些可能的类中的某一个。然而，更方便的方法是将其表示为一个向量$a_1, a_2, \cdots, a_k$，其中第i个元素（i就是真实的类）等于1，其他都等于0。可以将与此情况相关联的损失表示为一个依赖于向量p和向量a的损失函数。

一个常用于评估概率预测的标准就是二次损失函数（quadratic loss function）：

$$\sum_j (p_j - a_j)^2$$

这只是针对单个实例，这里的总和是所有可能的输出总和，而非不同的实例的总和。其中只有一个a值等于1，其余的a值都为0。因此，总和里面包括由不正确预测而得的p_j^2和由正确预测而得的$(1-p_i)^2$，从而公式可以转换为

158 $$1-2p_i+\sum_j p_j^2$$

这里i是正确的类。当测试集有多个实例时，损失函数便是所有单个实例损失函数的总和。

这里存在一个有趣的理论事实，当真实类的产生存在一定的概率性时，如果要使二次损失函数值最小化，最好的策略就是让向量p选择各类的真实概率，即p_i =Pr[class=i]。如果真实概率已知，它就是p的最佳选择。如果不知道，寻求最小二次损失值的系统则要力争得到对Pr[class = i]的最好估计并将其作为p_i值。

这点相当容易看到。真实概率表示为$p_1^*, p_2^*, \cdots, p_k^*$，则$p_i^* = \Pr[\text{class}=i]$。对一个测试实例的二次损失函数的期望值可以重新表达如下：

$$E\left[\sum_j (p_j-a_j)^2\right]=\sum_j \left(E\left[p_j^2\right]-2E\left[p_ja_j\right]+E\left[a_j^2\right]\right)$$
$$=\sum_j (p_j^2-2p_jp_j^*+p_j^*)=\sum_j ((p_j-p_j^*)+p_j^*(1-p_j^*))$$

第一步将期望符放到总和符号里面并将平方展开。第二步p_j是一个常数，a_j的期望值就是p_j^*，并且因a_j不是0就是1，所以$a_j^2=a_j$，a_j^2的期望值也等于p_j^*。第三步直接代数转换。要得到总和的最小值，很明显，就是使$p_j = p_j^*$，这样平方项消失，剩下的只有掌控实际类的真实分布的方差。

误差平方最小化在预测问题上已有很长的一段历史，二次损失函数促使预测器选择最好的概率估计，或者优先选择能对真实概率做出最好猜测的预测器。另外，二次损失函数有一些有用的理论属性，不在这里讨论。由于上述种种原因，二次损失函数常被当作评估概率预测成功的标准。

5.6.2 信息损失函数

另一个较为普遍的评估概率预测的标准就是信息损失函数(informational loss function)：

$$-\log_2 p_i$$

其中，第i个预测是正确预测。这实际上相当于一个负的对数似然函数（log-likelihood
159 function），是经logistic回归优化的函数，曾在4.6节中描述过。它用位数（bit）来代表所需信息量，这个信息量用来表示实际类i的概率分布$p_1, p_2, \cdots, p_k$。换句话说，如果已告诉你一个概

率分布、某人必须同你交流实际出现的应属哪一个类，就需要对信息进行编码，所需尽可能有效的位数（当然总是可以用更多的位数）。因概率总是小于1，它们的对数是负数，公式中的减号使结果成为正数。例如，一个二类问题即正面或是反面，每个类的概率是相等的，正面出现的位为1，因 $-\log_2 1/2$ 等于1。

如果真实概率是 $p_1^*, p_2^*, \cdots, p_k^*$，信息损失函数的期望值就是

$$-p_1^* \log_2 p_1 - p_2^* \log_2 p_2 - \cdots - p_k^* \log_2 p_k$$

如同二次损失函数，选择 $p_j = p_j^*$ 将表达式最小化，表达式变成真实分布的熵:

$$-p_1^* \log_2 p_1^* - p_2^* \log_2 p_2^* - \cdots - p_k^* \log_2 p_k^*$$

这样，信息损失函数能使知道真实概率的预测器做出真实的预测，并促使不知真实概率的预测器做出最好的猜测。

信息损失函数还可用赌博现象来解释。想象对测试结果进行赌博，给每个可能的类一个成败可能性，输赢是由类的最终出现结果决定的。连续的实例就像连续地下赌注，一个接一个地赢（或输），在整个测试集上的赢钱总额的对数就相当于信息损失函数值。在赌博中，预测尽可能准确的成败可能性是要付出代价的，从这个意义上说，真实（预测）也是要付出代价的。

信息损失函数的一个问题是如果赋予一个0概率给实际发生的事件，函数值就成负无穷大。好像在赌博中连衬衣都输了。无论事看起来有多肯定，谨慎的人从不在某个具体事件上下全注。同样地，谨慎的预测器在信息损失函数操作中从不赋予任何结果以0概率。当没有任何信息可以提供给这个结果作为预测基础时，便导致了一个问题称为零频率问题（zero-frequency problem），人们提出了不同的解决方案，如在（4.2节）朴素贝叶斯中讨论过的拉普拉斯估计器。

5.6.3 讨论

如果你在从事概率预测的评估工作，将使用两种损失函数中的哪一种呢？这是个好问题，没有一致意见的答案，这其实是个人喜好问题。它们都在做损失函数所期待的基本工作：最大限度地使预测器正确地预测真实概率。但是它们之间存在一些客观的区别，对选择会有参考价值。 160

二次损失函数不仅考虑了实际发生事件的概率，同时还考虑其他事件的概率。比如，一个四类的问题，假设将40%的概率赋予实际遇到的类，而将剩余的分配给其余的三个类。由于先前的二次损失函数表达式中出现的 p_j^2 总和使得如何分配将决定二次损失值。如果将剩余的60%均匀分配到另外三个类，达到的损失是最小的。不均匀的分配将使平方和增加。而另一方面，信息损失函数只依赖于对实际发生的事件所赋予的概率。如果你对某个即将发生的事件下了赌注，而且押对了，谁还在乎你其余的钱在其他事件上是如何分配的?

如果你对某一实际发生的事件赋予了很小的概率，信息损失函数会给你很大的惩罚。最大的惩罚便是对0概率，是无穷的。在赌博世界里对待错误的惩罚也是如此严厉！而二次损失函数属于比较温和的、被绑定在

$$1 + \sum_j p_j^2$$

绝对不会超出2。

最后，信息损失函数的提议者们提出一种对学习性能评估的一般理论，称为最短描述长度（MDL）原理（minimum description length principle）。他们主张一种方案所学的结构大小可以用信息的位值来衡量，如果损失值也用相同的单位来衡量，二者可以有效地结合在一起。我们将在5.9节中讨论。

5.7 计算成本

到目前为止，所讨论的评估方法都没有考虑到错误决策、错误分类的成本问题。不考虑错误成本的优化分类（成功）率经常会导致奇怪的结果。有个例子，机器学习曾用于判断奶牛场里的每一头母牛的确切动情期。识别母牛使用电子耳签，并使用各种不同的特性，
161 如产奶量及化学成分（由高科技的挤奶机器自动记录）、挤奶次序（母牛是有秩序的动物，通常是按相同的次序进挤奶棚的，除非是处于类似动情期这类非正常的情况）。在现代化的牧场手术中，事先知道母牛何时做好准备也很重要：动物的人工受精错过一个周期即导致不必要的延期下仔，使下一次更为复杂。在早期试验中，机器学习方案总是顽固地预测母牛始终没有处在动情期。和人类一样，母牛也有约隔30天左右的生理周期，因此这个“零”规则在约97%的时候都是正确的，这个正确率在任何农业领域都是令人难忘的！但是，我们所需要的当然是“处于动情期”的预测准确率高于“不在动情期”的准确率的分类规则：这两种误差的成本代价是不同的。使用分类正确率进行评估是在默认误差成本相同的假设前提下的。

其他反映不同误差成本的例子包括贷款决策，贷款给违规者的代价远高于由于拒绝贷款给不违规者而造成生意损失的代价。在海面浮油探测中，未能探测出威胁环境的海面浮油的错误成本远高于错误报警的代价。在负荷预报中，为防范一场实际未发生的暴风雪而调整发电机组所产生的成本远低于由于对暴风雪的袭击毫无防范所造成的损失。在诊断实例中，实际上没有问题的机器被误诊为有问题所产生的成本远小于即将导致机器失败的问题由于没有侦察到而造成的损失。在散发促销邮件时，散发垃圾邮件得不到回音所产生的成本远小于由于没有将邮件发送到会有回音的家庭而丧失生意机会所造成的损失。为什么所有第1章中列举的实例都在这里！事实上，你很难找到不同类型错误的成本是相同的实例。

对一个yes和no的二类问题，如借或不借，将可疑的斑点标记为浮油或不是浮油等等，一个预测可能产生四种不同的结果，见表5-3。正确的肯定（true positive, TP）和正确的否定（true negative, TN）都是正确的分类结果。错误的肯定（false positive, FP）发生在当预测结果为yes而实际上是no时，错误的否定（false negative, FN）发生在当预测结果为no而实际上是yes时。正确肯定率(true positive rate)是TP数值除以肯定类的总数TP + FN；错误肯定率(false positive rate)是FP数值除以否定类的总数FP + TN 。综合成功率是正确的分类数除以总体分类数：

$$\frac{TP+TN}{TP+TN+FP+FN}$$

最后，误差率就是用1减去它。

表5-3 二类预测的不同结果

		预测类	
		yes	no
真实类	yes	正确的肯定	错误的否定
	no	错误的肯定	正确的否定

在多类预测中，对测试集的预测结果经常使用二维混淆矩阵（confusion matrix）来展示，每个类都有对应的行和列。每个矩阵元显示的是测试样本数目，这些样本的真实类以对应的行显示为准，预测类则是对应的列所显示的类。好的结果是在主对角线上数值要大，而非主对角线（off-diagonal）元素的数值要小（理想情况为0)。表5-4(a)给出了一个三类问题的例子。在这个例子中，有200个测试集实例（矩阵中九个数字的总和)，其中88 + 40 + 12 = 140 是正确的预测，因此成功率是70%。

但这是公正的综合成功率衡量方法吗？有多少是偶然达成一致的？这个预测器预测120个测试样本属于*a*类、60个属于*b*类，20个属于c类。如果有一个随机预测器也达到如此相同的预测数目又会是怎样的呢？答案可在表5-4(b)中看到。表5-4(b)中第一行将100个实际属于*a*类的测试集按先前预测器的各类预测总体比例分摊，其余二类在第二行和第三行也照此法按比例分摊，这样得到的矩阵当然每行、每列的总和与先前的矩阵是相同的，实例数量没有改变，而且也保证了随机预测器和真实预测器对*a*、*b*、*c*三个类的预测数目是相等的。

表5-4 一个三类问题的不同预测结果：(a)真实的 (b)期望的

(a)

		预测类			总计
		a	b	c	
真实的类	a	88	10	2	100
	b	14	40	6	60
	c	18	10	12	40
	总计	120	60	20	

(b)

		预测类			总计
		a	b	c	
真实的类	a	60	30	10	100
	b	36	18	6	60
	c	24	12	4	40
	总计	120	60	20	

这个随机预测器使60 + 18 + 4 = 82个实例获得了正确的预测。一种衡量方法称为Kappa 统计量将这个期望值考虑进去，通过将其从预测器成功数中扣除，并将结果表示为它和一个完美的预测器所得结果的比例值，即从200 − 82 = 118个可能的成功数中得到140 − 82 = 58个额外的成功数目，或者说是49.2%。Kappa的最大值是100%，而具有相同列的随机预测器的Kappa期望值是0。总的来说，Kappa 统计量用于衡量对一个数据集预测分类和观察分类之间的一致性，一致性的修正偶尔发生。但是如同普通的成功率计算一样，它也没有考虑成本问题。 162~163

5.7.1 成本敏感分类

如果成本已知，它们可以被应用到决策过程的财务分析中。在一个二类问题中，混淆矩阵如表5-3，它的两种错误类型，错误的肯定和错误的否定，将会有不同的成本；同样地，两种不同的正确分类可能带来不同的收益。二类问题的成本可概括成一个2 × 2的矩阵，主对角元素代表了两种类型的正确分类，而非主对角元素代表了两种类型的错误分类。在多类情况

下，这就被推广为一个方阵，方阵大小即是类的个数，并且主对角元素也是代表了正确分类的成本。表5-5(a)、(b)分别展示了二类和三类缺省的成本矩阵，矩阵只是简单地给出了错误数目：每个错误分类成本都是1。

表5-5　缺省的成本矩阵：(a) 二类情况 (b) 三类情况

		预测类	
		yes	no
真实类	yes	0	1
	no	1	0
(a)			

		预测类		
		a	b	c
真实类	a	0	1	1
	b	1	0	1
	c	1	1	0
(b)				

考虑成本矩阵，用每个决策的平均成本（或者从正面看，考虑收益）来代替成功率。虽然在这里我们不这样做，但决策过程中完整的财务分析也许还要考虑使用机器学习工具的成
164 本，包括搜集训练集的成本和模型使用成本，或者是产生决策结构的成本，即决定测试实例属性的成本。如果所有这些成本都是已知的，成本矩阵中反映不同结果的数值可以估计出来，比如说用交叉验证法估计，那么进行这种财务分析便很简单了。

给出成本矩阵，可以计算某个具体学习模型在某个测试集上的成本，只要将模型对每个测试实例进行预测所形成的成本矩阵中的相关元素相加。进行预测时，成本将被忽略，但进行评估时则考虑成本。

如果模型能够输出与各个预测相关联的概率，便能调整期望预测成本到最小。给出对某个测试实例各个预测结果的概率，一般都会选择最有可能的那个预测结果。作为替换，模型也可以选择期望错误分类成本最低的那一个类作为预测结果。例如，假设有一个三类问题，分类模型赋予某一个测试实例属于a、b、c三个类的概率分别为p_a、p_b和p_c，它的成本矩阵如表5-5(b)。如果预测是属a类，并且这个预测结果是正确的，那么期望预测成本就是将矩阵的第一列[0,1,1]，和概率向量$[p_a, p_b, p_c]$相乘，得到p_b+p_c，或者$1-p_a$，因为三个概率的和等于1。类似地，另外两个类的预测成本分别是$1-p_b$和$1-p_c$。对这个成本矩阵来说，选择期望成本最低的预测就相当于选择了概率最大的类。对于不同的成本矩阵，情况可能会有所不同。

前提假设是学习方案能输出概率，就如同朴素贝叶斯方案那样。即使通常情况下不输出概率，大多数分类器还是很容易计算出概率的。如决策数，对一个测试实例的概率分布就是对应叶节点上的类分布。

5.7.2　成本敏感学习

我们已经看到怎样利用一个在建模时不考虑成本的分类器，做出对成本矩阵敏感的预测。在这种情况下，成本在训练阶段被忽略，但在预测阶段则要使用。另一种方法正好相反，在训练过程中考虑成本，而在预测时忽略成本。从理论上讲，如果学习算法给分类器合适的成本矩阵，可能会获得较好的性能。

对一个二类问题，有一个简单的常用方法能使任何一个学习方案变为成本敏感型。想法是生成拥有yes和no实例不同类别比例的训练数据。假设人为地提高数据集中属于no类的实例
165 数量到10倍，然后用这个数据集进行训练。如果学习方案是力争使错误数目最小化的，将形

成一个倾向于避免对no类实例错误分类的决策结构，因为这种错误会带来10倍的惩罚。如果在测试数据中，属于no类实例的比例还是按照它在原始数据中的比例，那么no类实例的错误将小于属于yes类的实例——也就是说错误的肯定将小于错误的否定——因为错误的肯定已被加权而达到10倍于错误的否定。改变训练集中实例的比例是一种建立成本敏感分类器的常用技术。

改变训练集实例比例的一种方法是复制数据集中实例。另一方面，很多学习方案允许对实例加权。（正如我们在3.2节中提到的，这是处理残缺数值的一种常用技术。）实例的权通常初始为1。为了建立成本敏感的树，权可被初始为两种错误的相对成本，即错误的肯定和错误的否定所对应的成本。

5.7.3 上升图

在现实中，很难知道成本的正确度，人们要考虑各种不同的场景。想象一下你在从事直接邮寄广告的业务，要将一个促销广告大规模邮寄给1 000 000户家庭，当然大部分的家庭是不会有回应的。根据以往的经验，得到回应的比例是0.1%（有1000个回应者）。假设我们现有一套数据挖掘工具，根据我们对这些家庭的掌握信息，能识别出其中的100 000户家庭子集，回应率达0.4%（有400个回应者）。根据邮寄成本与回应所带来盈利的比较，将邮寄广告限定在这100 000户家庭上许是很划算的。在市场学术语中，回应率的增值，在这个例子中为系数4，称为上升系数（lift factor），由学习工具得到。如果知道成本，就能确定某个具体上升系数所隐含的盈利。

然而，你很可能还需要评估其他的可能性。采用相同的数据挖掘方案而选定不同参数设定，也许会识别出其中的400 000户家庭，回应率达0.2%（有800个回应者），对应的上升系数为2。同样地，这是否邮寄广告更有利可图的目标，可以从所投入的成本计算中看出。考虑模型建立和模型应用成本因素也许是必要的，这包括生成属性值所需的信息搜集。毕竟，如果模型生成代价非常昂贵，大规模的邮寄比锁定目标更具成本效率。

给定一个学习方案，输出对测试数据集每个成员的类预测概率（如朴素贝叶斯方案），找出一些测试实例子集，这些子集所含的肯定类实例的比例要高于肯定类在整个测试集中所占的比例。为此，将所有测试实例按照预测yes概率的降序排列。这样，要找出一个大小给定、肯定类实例所占比例尽可能大的样本，只要在测试实例序列中从头开始读取要求数量的实例。 166
如果每个测试实例的真实类已知，便可计算上升系数，只要将样本中肯定类实例数量除以样本数量得到一个成功比例，然后再除以整个测试集的成功比例，得到上升系数。

表5-6给出一个例子，一个小数据集含150个实例，其中50个是yes回应，因此总体成功比例是33%。所有的实例已经按照它们被预测为yes回应概率的降序排列。序列中的第一个实例是学习方法认为最有可能得到肯定回应的，第二个是下一个最有可能的，依次类推。概率的数值并不重要：排名才是重要的。每个实例的真实类排名都已给出。可以看出，这个学习方法对第一项和第二项的预测都是正确的，它们确实是肯定的。可是第三项却是错误的，它的结果是否定的。如果你现在要寻找10个最有希望的实例，但只知道预测概率而不知真实的类，最佳的选择就是排名中的前十个实例。其中8个是有肯定回应的，因此成功比例就是80%，对应的上升系数约为2.4。

表5-6 上升图数据

排名	预测概率	实际类	排名	预测概率	实际类
1	0.95	yes	11	0.77	no
2	0.93	yes	12	0.76	yes
3	0.93	no	13	0.73	yes
4	0.88	yes	14	0.65	no
5	0.86	yes	15	0.63	yes
6	0.85	yes	16	0.58	no
7	0.82	yes	17	0.56	yes
8	0.80	yes	18	0.49	no
9	0.80	no	19	0.48	yes
10	0.79	yes	…	…	…

如果你知道所投入的不同成本，就可以计算出每种样本的大小，然后选择最有收益的样本。然而，通过图形展示各种不同的可能性经常比只提供一个“最佳”决策更有启迪作用。用不同大小的样本重复上述操作，便能画出如图5-1的上升图。横轴是样本大小与所有可能邮寄数量的比例。纵轴是所得到的回应数量。左下角点和右上角点分别代表没有邮寄得到0个回应和全数邮寄得到1000个回应。对角线给出了针对不同大小的随机样本的期望结果。但我们
167 不是随机抽取样本，而是利用数据挖掘工具选择那些最有可能给出回应的实例样本。这在图中对应位于上方的那根曲线，它是由实际得到的回应数之和与对应的按概率排序的实例百分比来确定的。上面讨论过的两个具体例子在图中标记为：10%邮寄比例产生400个回应者和40%邮寄比例产生800个回应者。

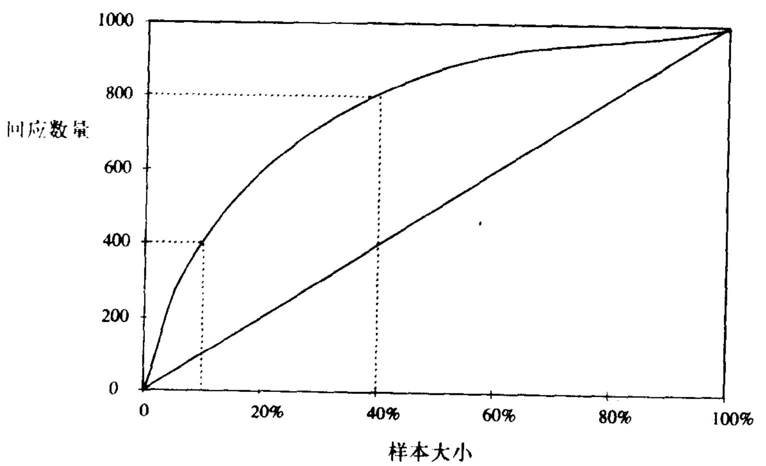


图5-1 一个假设的上升图

你所希望的位置是在图中靠近左上角，最好的情况是1000个邮寄广告获得1000个回应，这时你只将邮件寄给会有回应的家庭并获得100%的成功率。任何值得命名的选择程序都会让你保持在对角线上方，否则你的结果比随机取样还差。因此，上升图的工作部位是在上三角位置，越往西北方向越好。

5.7.4 ROC曲线

上升图是有用的工具，广泛应用于市场营销领域。它和一种评估数据挖掘方案的图形技

术ROC曲线关系密切。ROC曲线同样适用于上述情形，学习器选择测试实例样本时尽量使肯定结果比例较高。“接受者操作特性（receiver operating characteristic）”的缩写（即ROC），是一种用于信号探测的术语，用来体现噪声信道击中率和错误报警之间的平衡。ROC 曲线描绘分类器的性能而不考虑类分布或误差成本。纵轴是样本中所含肯定类的数量，表示为它在肯定类总数中所占的百分比，横轴是样本中所含否定类的数量，表示为它在否定类总数中所占的百分比。纵轴实际上和上升图是一样的，只是它是用百分比来表示。横轴稍许有点差别，由否定类数量代替样本大小。然而，在市场直销情形中，肯定结果的比例是非常小的（如0.1%），样本大小和否定类数量之间的差别可以忽略，因此ROC 曲线和上升图看起来很相似。与上升图相似，左上角是工作位置。 168

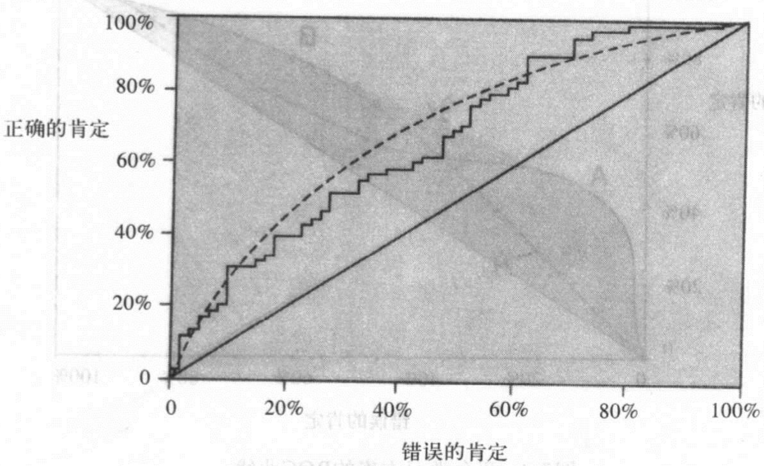


图5-2 ROC 曲线

图5-2 展示了表5-6所列测试数据样本的 ROC曲线（锯齿线）。你可以和表结合在一起看。从原点开始，向上2（2个肯定类），向右1（1个否定类），向上5（5个肯定类），向右1（1个否定类），向上1，向右1，向上2等等。每个点对应画出序列表中的某个位置，累计yes 和no 的数量分别朝纵向和横向画线。顺着序列表往下，样本扩大，肯定类和否定类数量也随之增长。

图5-2 中的锯齿形ROC线依赖于具体测试数据样本内容。运用交叉验证可以降低对样本的依赖。对每个不同的no类数量，即沿横轴方向的每个位置，取适量排在前面的实例使no类数量正好达到，计算其中yes类个数，最后将交叉验证的不同折所得的yes 类个数取平均值。结果得到如图5-2中的光滑曲线，但在现实中这种曲线并不如此光滑。 169

这只是利用交叉验证产生ROC曲线的一种方法，更简便的方法是收集在所有不同测试集（在10折交叉验证中存在10个）上的预测概率，连同各个实例所对应的真实类标签，然后根据这些数据生成一个排名表。这里假设由不同训练集建立的分类器的概率估计都是基于相同大小的随机数据样本。到底哪种方法更好并不很清楚，但是后者更易实现。

如果学习方案不能对实例进行排序，可以如前所述先将其变为成本敏感型。在10折交叉验证的每个折中，对实例选择不同的成本比例加权，在加了权的数据集上进行方案训练，在测试集上计算正确的肯定和错误的肯定数量，然后将结果画在ROC轴上。（测试集是否加权没有关系，因为在ROC图中坐标轴数值是用正确或错误的肯定的百分比来表示的。）但是，对一些原本就成本敏感的概率性分类器（如朴素贝叶斯）来说，成本大大增加，因为在曲线的每

个点上都要包含一个独立的学习问题。

比较一下由不同的学习方案经交叉验证的ROC 曲线是很有帮助的。例如，图5-3中，如果要寻求一个规模较小且集中的样本，也就是说如果图形左侧是你的工作位置，那么A模型有优越性。明确地说，如果你的目标是要覆盖40%正确的肯定，就选择A模型。其错误的肯定率在5% 左右，这比错误的肯定率达20%以上的B模型要好。但如果你计划使用一个大型的样本，B模型具有优越性。如果你要覆盖80%正确的肯定，B模型所产生的错误的肯定率在60%，而A模型则达80%。阴影区域称为两个曲线之间的凸包（convex hull），你总是要工作在凸包的上边界。

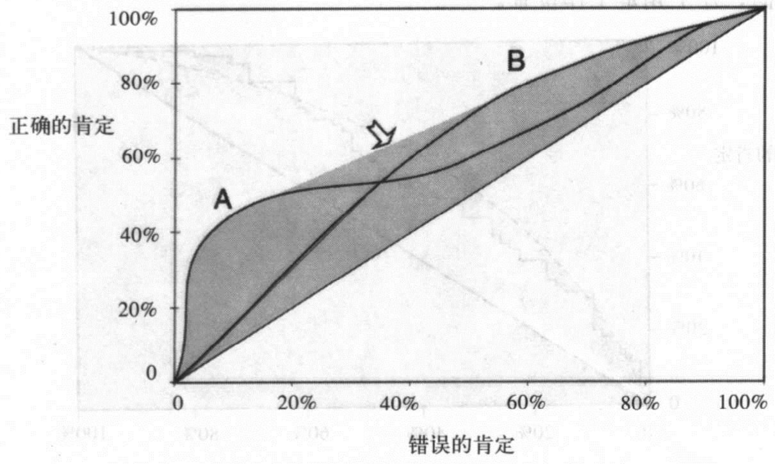


图5-3 两个学习方案的ROC曲线

那么凸包上边界的中间区域，既不属于A方案也不属于B方案，该如何看待呢？这是个值得注意的事实，你可以将A方案和B方案以适当的概率随机组合，而达到阴影区域的任何位置。为了看到这点，我们为正确和错误的肯定率分别达到t_A和f_A的A方案选择某一具体的概率，正确和错误的肯定率分别达到t_B和f_B的B方案选择另一个概率。如果你随意使用这两个学习方案的概率分别为p和q，且$p+q = 1$，那么你将获得正确和错误的肯定率分别为$p \cdot t_A + q \cdot t_B$和$p \cdot f_A + q \cdot f_B$。这代表了位于$(t_A, f_A)$和$(t_B, f_B)$两点之间的直线上的某一个点，改变$p$和$q$的值可以描绘出这两点之间的整条连线。这样整个阴影部分都可以获得。只有当某个方案产生一个点正好落在凸包上边界，要单独使用这个方案；否则的话，组合使用分类器与凸包上某点对应总会好一些。

5.7.5 反馈率-精确率曲线

人们已经通过描绘各个不同领域的上升图和ROC曲线，抓住了一些基本性的权衡问题。信息检索便是一个好例子。提交一个查询，网络搜索引擎即列出一串经推测与问题相关的文件。某个系统有100个文件，其中40个为相关文件，而另一系统有400个文件，其中80个相关文件。比较两个系统，哪个更好呢？答案很明显，它是由错误的肯定，即被返回但却不相关的文件数，以及错误的否定，即相关的却没有被返回的文件数所对应的成本决定的。信息检索研究者定义了两个参数，称为反馈率（recall）和精确率（precision）:

$$反馈率=\frac{检索到的相关文件数量}{相关文件总数量}$$

$$精确率=\frac{检索到的相关文件数量}{检索到的文件总数量}$$

例如，如果表5-6中所列的是检索反馈的文件排名，yes 和no 表示这个文件是否相关，而整个收藏集中共包含了40个相关的文件，那么“10项反馈率”就是指排名中前十项的反馈率，即是8/40＝20%；“10项精确率”就是8/10＝80%。信息检索专家使用反馈率-精确率曲线，如同ROC曲线和上升图一样，对不同的反馈文件数目，画出相应的反馈率和精确率。只是因坐标轴不同，曲线成双曲形状，期望的工作点在右上角。 170~171

表5-7　权衡错误肯定和错误否定的不同评估度量方法

	领　域	绘　制	轴	轴的含义
上升图	市场营销	正确的肯定(TP)	TP	正确的肯定数量
		对应子集容量	子集容量	$\frac{TP+FP}{TP+FP+TN+FN}\times 100\%$
ROC曲线	通信	正确肯定率(TP)	TP率	$tp=\frac{TP}{TP+FN}\times 100\%$
		对应错误肯定率(FP)	FP率	$fp=\frac{FP}{FP+TN}\times 100\%$
反馈率-精确率曲线	信息检索	反馈率	反馈率	同以上的TP率tp
		对应精确率	精确率	$\frac{TP}{TP+FP}\times 100\%$

5.7.6　讨论

表5-7总结了我们已介绍的三个基本权衡的不同评估方法，TP，FP，TN和FN分别代表正确的肯定、错误的肯定、正确的否定和错误的否定。你想选择一个实例集，包含yes类实例的比例较高，并且yes类的覆盖量也较高的，可以（保守地）选用略小一点的覆盖量提高类比例，或者损失类比例（公平地）提高覆盖量。不同的技术提供不同的折衷方案，并可用上述任何一种图形绘制出不同的曲线。

人们还试图寻找单一的量度来体现性能。在信息检索中使用的两种方法就是3点平均反馈率（three-point average recall），即在反馈率达20%、50%和80%时，所对应的三个精确率的平均值；另一种是11点平均反馈率，即在反馈率达0%、10%、20%、30%、40%、50%、60%、70%、80%、90%和100%时所对应的精确率的平均值。F测量（F-measure）用于信息检索，即：

$$\frac{2\times 反馈率\times 精确率}{反馈率+精确率}=\frac{2\cdot TP}{2\cdot TP+FP+FN}$$ 172

不同的领域使用不同的术语。例如在医学上谈论的诊断试验的敏感性和特异性(specificity)。敏感性是指在患病人群中，诊断结果是肯定（得病）的比例，即tp。特异性是指在没有病的人群中，诊断结果也是否定的比例，即$1-fp$。有时它们的乘积被当作是一种总体衡量：

$$\text{敏感性} \times \text{特异性} = tp\,(1-fp) = \frac{\text{TP}\cdot\text{TN}}{(\text{TP}+\text{FN})\cdot(\text{FP}+\text{TN})}$$

最后，当然还有我们的老朋友，成功率：

$$\frac{\text{TP}+\text{TN}}{\text{TP}+\text{FP}+\text{TN}+\text{FN}}$$

为了将ROC曲线概括成单一的数量，人们有时选用曲线下方的面积（AUC），因为一般说来，面积越大，模型越好。这个面积也可以很恰当地解释为分类器将任意抽取的肯定类实例排列在任意抽取的否定类实例之前的概率。如果在成本和类分布都是未知的情形下，要选择一种方案来处理所有的情况，这种度量方法也许会有帮助，但是仅靠一个数字是无法尽善尽美地俘获一个折衷问题的。这也只适合类似于上升图、ROC曲线和反馈率－精确率那样的二维描述。

5.7.7 成本曲线

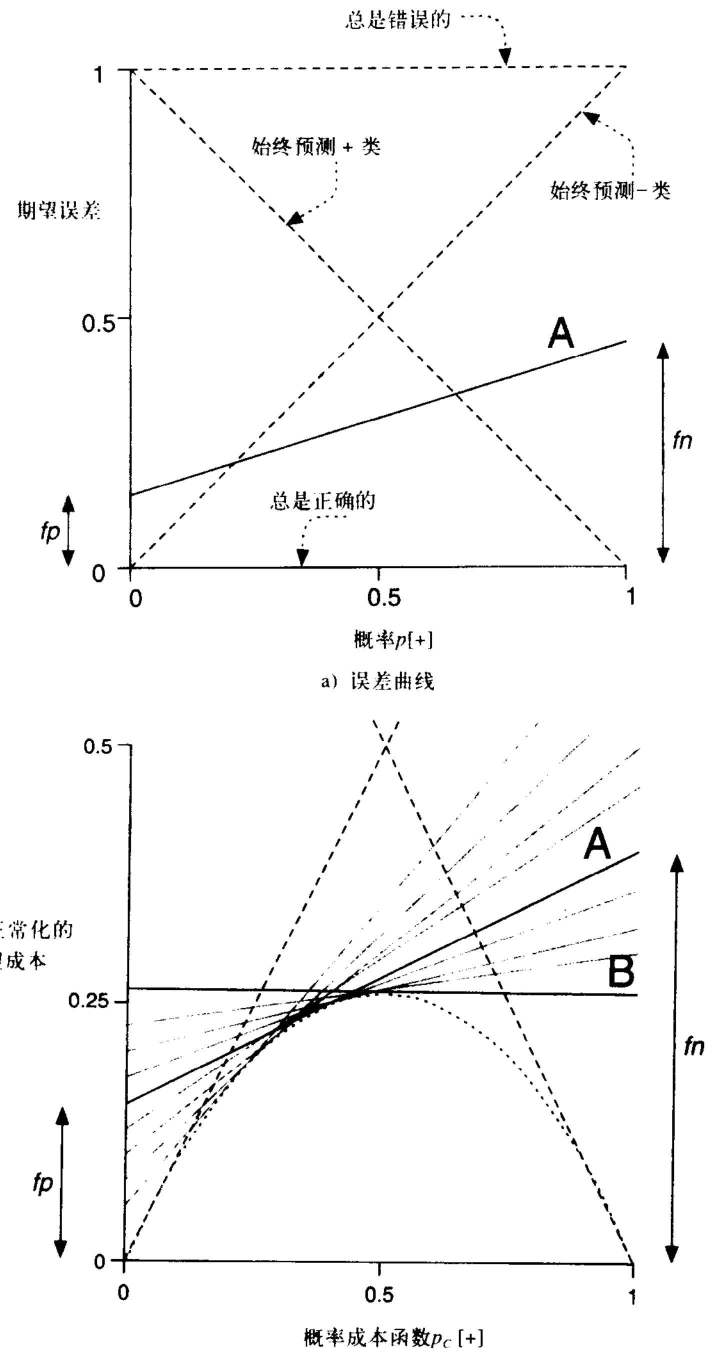
ROC曲线以及与此类似的其他方法在探测不同分类器及其成本之间的权衡是非常有用的。但是它们的不理想之处是要在已知错误成本的情形下评估机器学习模型。比如要读出某个分类器的期望成本，即一个确定的成本矩阵和类分布，并不是件容易的事。要确定不同分类器的适用性也不容易。图5-3中，在两条ROC曲线交叉点，很难说分类器A以多大的成本和类分布胜过分类器B。

成本曲线是另一种不同的展示方法，一个分类器对应一条直线，它所展示的是性能如何随类分布的变化而变化。它们也是针对二类问题效果最佳，尽管多类问题总是可以通过挑选出其中一个类对照其余的类而转化为二类问题。

图5-4a画出了期望误差与其中一个类概率之间的关系。你可以想象通过对测试集不均匀重新取样来调整这个概率。我们将两个类分别表示为＋和－，两条对角线展示的是两个极端
173 的分类器的性能表现：其中一个分类器始终预测是＋类，如果数据集中没有属于＋类的实例，那么期望误差则为1，如果所有实例都属于＋类，那么期望误差则为0；另一种分类器始终预测是－类，则给出正好相反的性能。水平虚线表示分类器预测始终是错误的，而X轴本身则表示分类器预测始终是正确的。当然这些在实践中都是不现实的。好的分类器误差率低，因此你所想要的是尽可能地接近图形底部。

直线A代表了某个具体的分类器的误差率。如果你在某个测试集上计算它的性能表现，错误肯定率fp便是它在全部属于否定类（$p[+]=0$）的测试子集上的期望误差率，而错误否定率fn便是它在全部属于肯定类（$p[+]=1$）的测试子集上的误差率。它们的值分别是直线左右两端的纵坐标值。从图中你即刻可以发现当$p[+]$小于0.2时，始终预测－类的极端分类器性能优于预测A，而当$p[+]$大于约0.65时，另一个极端分类器性能优于预测器A。

到目前为止，我们还没有考虑成本问题，或者可以说用的是缺省成本矩阵，所有的错误所产生的成本是一样的。考虑误差成本的成本曲线，除了坐标轴不同外，看起来非常地相似。图5-4b展示了分类器A的成本曲线（注意为方便起见，纵坐标比例放大了。我们现在暂时忽略图中的灰线条）。成本曲线图展示的是使用A的期望成本对应概率成本函数。概率成本函数其实是$p[+]$的变形，同样保持着两种极端：当$p[+]=0$时期望成本为0，当$p[+]=1$时则为1。当实例预测值是+类而实属－类时的成本表示为$C[+|-]$，反之则表示为$C[-|+]$。这样，图5-4b的坐标轴分别是



a）误差曲线

b）成本曲线

图5-4 概率变化所产生的影响

$$经正常化的期望成本=fn \times p_C[+]+fp \times (1-p_C[+])$$

$$概率成本函数\ p_C[+]=\frac{p[+]C[+|-]}{p[+]C[+|-]+p[-]C[-|+]}$$

174~175 我们假设正确的预测是没有成本耗费的，即$C[+|+]=C[-|-]=0$。如果不是这样，上述公式则还要复杂一些。

经正常化的期望成本最大值可达到1，这也是为什么称它是“经正常化”的原因。成本曲线好在如同误差曲线一样，图中的左右极端成本值正是fp和fn，因此可以很容易地画出任意一个分类器的成本曲线图。

图5-4b中分类器B的期望成本始终保持一致，也就是说它的错误肯定和错误否定率是相等的。正如你会看到的，当概率成本函数值约大于0.45时，它的性能优于分类器A，知道了成本，我们便能很容易地计算出相应的类分布。遇到有不同类分布的情形，用成本曲线可很容易地表明某个分类器性能优于另一个。

在何种情况下这会有用呢？回到我们讲述的预测母牛动情期的例子，它们有30天的周期，或者说1/30的先验概率，一般不会有很大的变化（除非基因突变！）。但某个牧场在任何指定的星期里很可能处于动情期的母牛会有不同的比例，也许和月亮的月相同步，谁知道呢？因此，不同的时期适合使用不同的分类器。在油污溢出实例中，不同批的数据会得到不同的溢出概率。在这些情况下，成本曲线可帮助用来显示何时使用何种分类器。

在上升图、ROC曲线或反馈率–精确率曲线上，每一个点代表一个由某种算法（如朴素贝叶斯算法）取不同阈值所获得的分类器。成本曲线中每个分类器由一条直线来代表，一系列的分类器将扫出一条弯曲的分类器包络线，它的下限显示出这种类型的分类器当参数选择适当时性能表现将如何好。图5-4b用一系列的灰线条展示了这些。如果继续下去，最终将扫出图中的那条抛物线点虚线。

分类器B的工作区间在概率成本值约0.25到0.75之间。在这个区间之外，由虚线表示的其他分类器性能比B要好。假设，我们决定要在这个工作区中使用分类器B并在这个工作区之上或之下的范围使用其他合适分类器。所有在抛物线上的点肯定都比这个方案要好，但究竟有多好？从ROC曲线的角度很难回答这个问题，但从成本曲线来看就很容易解答。当概率成本值在0.5左右时，性能差异可忽略不计，当小于0.2或大于0.8时，也几乎看不到差异。最大的差异发生在概率成本值为0.25和0.75时，差异约有0.04，即最大可能成本值的4%。

5.8 评估数值预测

我们目前所讨论的评估方法都是有关分类预测问题的，而不是数值预测问题。要使用独
176 立于训练集之外的测试集，如旁置法、交叉验证法来做性能评估，这个基本原理同样适用于数值预测。而由计算误差率所提供的基本定性方面的量度已不再适合：错误不再是简单的有或没有的问题，错误会呈现不同的大小。

表5-8中总结了几种方法，可用来评估数值预测成功与否。对测试实例的预测值为$p_1, p_2, \cdots, p_n$，真实值为$a_1, a_2, \cdots, a_n$。注意p_i在这里与上一节中提到的有非常大的区别：上节中是指预测结果为第i类的概率，这里是指对第i个测试实例的预测值。

均方误差（mean-squared error）是最常用的基本方法，有时再取其平方根获得与预测值一致的尺度。在许多数学技术中（如第4章中讨论的线性回归）都应用均方误差，因为在数学处理上它似乎是最容易方法，正如数学家们说的“循规蹈矩”。但是，这里我们把它当作是一种性能量度，而所有的性能量度都是易于计算的，因此均方误差没有什么优势。问题是它是否是合适的量度方法呢？

平均绝对误差（mean absolute error）是另一种可供选择的方法：将各个误差的大小取平均值，不考虑它们的正负符号。均方误差趋向夸大外围物（outlier），即预测误差比其他大的实例的影响，而绝对误差没有这方面的影响，对所有的错误，根据它们的大小公平对待。

有时相对误差比绝对误差值更重要。举例来说，如果说10%的误差率无论对于预测值为500而其中误差值50，还是对于预测值为2而其中误差值0.2的情况下是同样重要的，那么绝对误差的平均值就毫无意义了，而考虑相对误差比较适当。可以在计算均方误差或平均绝对误差中使用相对误差而将这个问题考虑进去。

表5-8中的相对平方误差（Relative squared error）含义有些特别。这个误差是由预测器和另一个简单的预测器相比较而得到的。这里所指的简单预测器只是训练集真实数值的平均值。因此相对平方误差是将该预测器总的平方误差正常化，即除以缺省预测器总的平方误差。

下一个测量误差称为相对绝对误差（Relative absolute error），其实只是将总的绝对误差同样进行正常化。这三个相对误差测量方法都是使用预测平均值的简单预测器误差进行正常化。

表5-8 数值预测的性能衡量

性能衡量	公式
均方误差	$\dfrac{(p_1-a_1)^2+\cdots+(p_n-a_n)^2}{n}$
均方根误差	$\sqrt{\dfrac{(p_1-a_1)^2+\cdots+(p_n-a_n)^2}{n}}$
平均绝对误差	$\dfrac{\lvert p_1-a_1\rvert+\cdots+\lvert p_n-a_n\rvert}{n}$
相对平方误差	$\dfrac{(p_1-a_1)^2+\cdots+(p_n-a_n)^2}{(a_1-\bar{a})^2+\cdots+(a_n-\bar{a})^2}$ 其中 $\bar{a}=\frac{1}{n}\sum_i a_i$
相对平方根误差	$\sqrt{\dfrac{(p_1-a_1)^2+\cdots+(p_n-a_n)^2}{(a_1-\bar{a})^2+\cdots+(a_n-\bar{a})^2}}$
相对绝对误差	$\dfrac{\lvert p_1-a_1\rvert+\cdots+\lvert p_n-a_n\rvert}{\lvert a_1-\bar{a}\rvert+\cdots+\lvert a_n-\bar{a}\rvert}$
相关系数	$\dfrac{S_{PA}}{\sqrt{S_P S_A}}$ 其中 $S_{PA}=\dfrac{\sum_i(p_i-\bar{p})(a_i-\bar{a})}{n-1}$ $S_p=\dfrac{\sum_i(p_i-\bar{p})^2}{n-1}$ 和 $S_A=\dfrac{\sum_i(a_i-\bar{a})^2}{n-1}$

注：p为预测值，a为真实值

表5-8 中的最后一项是相关系数（Correlation coefficient），它是测量真实值a和预测值p之间的统计相关性。相关系数值从完全相关时等于1，到完全不相关时等于0，再到反向完全相关时等于 -1。当然，对一个合理的预测方法来说是不会出现负数的。相关性和其他的测量方法比较起来有些不同，因为它是量度独立的，如果你得到一组具体预测值，当真实值不变而所有的预测值都乘以某个常因子时，它的相关系数是不会改变的。这个常因子在分子S_{PA}的每一项中出现，也在分母S_P的每一项中出现，因此可以相互抵消。（这个特性在相对误差中不成立，如果你将所有的预测值都乘以一个较大的常数，那将使预测值和真实值的差值明显变化，

从而使误差百分率也明显变化。）另一个不同点在于，好性能的预测器其相关系数较大，而用其他方法计算误差率时，好性能的预测器误差值较小。

表5-9 四种数值预测模型的性能衡量

	A	B	C	D
均方根误差	67.8	91.7	63.3	57.4
平均绝对误差	41.3	38.5	33.4	29.2
相对平方根误差	42.2%	57.2%	39.4%	35.8%
相对绝对误差	43.1%	40.1%	34.8%	30.4%
相关系数	0.88	0.88	0.89	0.91

对于一种给定情形，使用何种测量方法比较合适，只有通过分析各种应用本身来决定。我们试图使什么达到最小化？不同误差类型的成本是多少？这些经常都是不易决定的。平方误差和平方根误差测量方法对大差异加权要比小差异重得多，而绝对误差不是这样。取平方根（均方根误差）的方法只是降低量度单位等级使其与预测值一致。相对误差试图均衡输出变量基本的可预知性或不可预知性，如果预测值相当接近平均值，那么认为预测较好，相对误差值均衡了这一点。否则，如果误差在一种情形下比在另一种情形下大很多，也许是因为在前一种情形下数值原本变化幅度较大，因此很难预测，而并非是预测器性能变差了。

值得庆幸的是，在大多数实际情形中，最好的数值预测方法无论使用何种误差测量法结果都是最好的。例如，表5-9列出了四种不同的数值预测技术在某个数据集上，采用交叉验证法的测试结果。根据5种误差计算结果，方法D是最好的：所有误差测量法的误差值最小，相关系数最大。方法C列第二位。方法A和B的性能是有争议的：它们的相关系数相等，从均方误差和相对平方误差来看，A比B好；但从绝对误差和相对绝对误差来看，结论正相反。这个差异或许是为了特别强调外围物应用了平方运算而产生的。

当比较两种不同数值预测的学习方案时，5.5节中所述的方法也是可行的。差别只在进行显著性试验（significance test）时，成功率被适当的性能衡量（如均方根误差）所替代了。

5.9 最短描述长度原理

177～179 机器学习方案所学到的是样本所描述的关于某个领域的一种“理论”，这个理论是有预测力的，能揭示有关这个领域的新的事实，换个说法就是能预测未知实例的类。理论是一个相当宏大的术语：在这里我们只是指一个有预测力的模型，因此理论有可能由决策树或一系列规则所组成，它们不必比这更理论化。

长期以来，科学界一贯认为在其他条件相同的情况下，简单的理论比复杂的理论更可取。这便是继中世纪哲学家William. Occam之后著名的Occam剃刀。Occam剃刀剃去了理论的哲学毛发。它的观点是：最好的科学理论应是最简单的，但能解释所有的事实真相。正如爱因斯坦的一句名言，“所有东西都要尽可能地简单化，没有更加简单的。”当然，在“其他条件相同”的背后隐藏着许多东西，很难客观地评估某个具体理论是否真的能“解释”它所基于的所有事实真相，这便是科学界的争议所在。

在机器学习中，大多数理论都有误差。如果说所学的是理论，那么所犯的错误就如同是这个理论的例外。一种确保其他条件相同的方法就是要强调当判定理论的“简单性”时，例外所包含的信息是理论的一部分。

想象一条并不完美的理论，它有几个例外。这个理论不能解释所有的数据，但可以解释大部分。我们所要做的就是将例外附加在理论中，清楚地说明这些是例外。新的理论变大了：这是代价，非常公平，是为它没有能力解释所有数据而付出的。然而，与全面而准确的复杂理论相比，原始理论的简单性（称之为简练是否有些过奖了？）也许足以弥补它不能解释所有东西的缺陷。

例如，开普勒（Kepler）在那个时代提出的三条行星运行规律不如哥白尼对托勒密（Ptolemaic）本轮理论的最后加细那样，能贴切地解释所有已知数据，它们的优势却在于复杂度很低，这便弥补了它的些许不准确。开普勒很清楚简洁理论的益处，尽管这个理论违背了他自己的唯美观点，它是基于"椭圆"而非完美的圆周运动。他曾用了一个很有个性的比喻："我清除了宇宙循环和旋转中最脏的部分，在我身后留下的只是一车垃圾。"

最短描述长度（mininum description length）或称MDL原理是指对于一堆数据来说，最好的理论是最小化使理论本身大小加上用于说明相关例外所需信息量，即最小的一车垃圾。在统计估计理论中，已非常成功地应用在各种参数拟合上。在机器学习中应用如下：给定一组 180
实例，使用一个学习方案从这些实例中推出一条理论，一条很简洁的，也许不配称为理论。这里借用通讯的比喻，想象这些实例正在一条毫无干扰的信道中传输信息，探测到的任何相似之处都被挖掘出来概括为更加压缩的编码。根据MDL原理，最好的概括便是使进行概括通讯所需的位数以及概括所基于的样本数达到最小值。

现在和5.6节中介绍的信息损失函数联系起来。信息损失函数用传输实例所需的位数来衡量误差，由理论给出概率预测。根据MDL我们还需要加入经过适当编码的理论"规模"位值，得到一个反映理论复杂度的综合数值。MDL原理引用了传输样本所需的信息，这里样本是指理论形成所基于的样本，即训练集实例而非测试集实例。这时过度拟合问题将被避免，因为相对于简单理论来说，一个复杂理论由于需要较多的位数来进行编码，将受到拟合的惩罚。一种极端是一个非常复杂、极过度拟合的理论在训练集上没有任何错误。另一种极端是一个非常简单的理论（零理论（null theory），它对训练集的传输一点帮助都没有。存在于这两者之间的是中等复杂度的理论，做出的概率预测不是很完美，需要通过传输一些有关训练集的信息来纠正。MDL原理提供了一种比较所有可能的理论的方法，从统一的立足点来看哪一种是最好的。我们找到了圣杯：一种只用训练集而不需独立的测试集的评估方案。可是也有困难之处，我们将在后面看到。

假设一个学习方案基于训练集E的样本得出一理论T，理论编码所需位数为$L[T]$（L代表长度）。给定理论、训练集本身编码位数为$L[E|T]$。$L[E|T]$实际上是由所有训练集成员信息损失函数值总和所给定。那么理论和训练集描述长度总和为

$$L[T]+L[E|T]$$

MDL原理建议采用使总和达到最小值的理论T。

MDL原理和基本概率理论之间存在着一个显著的联系。给定一个训练集E，我们寻找"可能性最大"的理论T，即寻找能使后验概率（posteriori probability）$\Pr[T|E]$即样本出现后的概率最大化的理论。如同在4.2节中所见的贝叶斯规则的条件概率（conditional probability），指出

$$\Pr[T|E]=\frac{\Pr[E|T]\Pr[T]}{\Pr[E]}$$ 181

取其负对数，

$$-\log\Pr[T \mid E] = -\log\Pr[E \mid T] - \log\Pr[T] + \log\Pr[E]$$

求概率的最大值相当于求其负对数的最小值。现在（如在5.6节中所见）编码所需的位数就是取其概率的负对数。而且，最后一项log Pr[E]只取决于训练集而与学习方案无关。因此，选择使概率Pr[$T \mid E$]达到最大值的理论等价于选择使

$$L[E|T] + L[T]$$

达到最小值的理论。换句话说，就是等价于MDL原理！

考虑训练集的理论后验概率并使其最大化的观点与MDL原理有如此令人诧异的关联，更增添了我们对最短描述长度原理的信任。但从中也指出了在实践中运用MDL的问题所在。直接应用贝叶斯理论的困难在于如何得到理论的恰当先验概率分布Pr[T]。在MDL公式中，问题转换为如何用最有效的方法将理论T编码成位值表示。编码有很多方法，编码和解码都要依赖于某个共同的先决假设，如果你事先知道理论将采用何种形式，你便能有效地利用信息进行编码。究竟怎样对理论T进行编码呢？困难随着对问题细节的深入出现了。

根据T，对E进行编码得到L[$E \mid T$]比较直接。先前已遇到过信息损失函数，但实际上当你对训练集中的每个成员依次编码，你是在进行一种序列编码而非对一个数据集编码。没有必要将训练集按一定的顺序传送，考虑这点应当有可能减少所需的位数。常用的近似方法是简单地减去log $n!$（这里n是训练集E所含成员数量），这是对训练集按某特定排列进行说明所需的位数（这个值对所有的理论来说都是一样的，实际上并不影响不同理论之间进行比较）。但是，人们可以想象利用个体误差频率来减少对误差的编码的位数。当然，误差编码使用的方法越是精密，对理论编码的精密要求就越低。因此，判断一条理论是否合理，在部分程度上取决于如何对误差进行编码。注重细节，并非想像中那么简单。

在这里我们不再深入讨论不同编码方法的细节。MDL方法单单基于训练集数据来评估学
182 习方案的问题是一块活跃的研究领域，对此研究者们持有不同的意见。

如这节的开头那样，我们还用哲学观点来做结尾。非常感谢Occam剃刀，简洁理论优先于复杂理论，站在哲学角度说是个公理，而不是能由基本理论推证出来的。这是我们所受的教育以及所处的时代的产物，似乎是不证自明的。简洁为先是（或也许是）文化偏好，而非绝对的。

希腊哲学家伊壁鸠鲁（Epicurus, 喜好美食、美酒，提倡享受世俗快乐，当然不是过分的）提出了几乎相反的观点。他的多种解释原理（principle of multiple explanations）建议“如果多种理论都与数据相符，保留这些理论”，他的基本观点是如果几种解释是一致的，考虑所有的解释也许能得到更精确的结论。总之，武断地丢弃任何解释都是不科学的。这种观点引出了基于实例的学习方法，保留所有的证据提供预测，然后使用组合决策的方法，如装袋（bagging）和提升（boosting）（将在第7章中讨论），它们确实是靠组合多种解释来获得预测力的。

5.10 聚类方法中应用MDL原理

最短描述长度原理的好处之一是不像其他的评估标准，可适用于各种完全不同的情况。虽然如我们先前所见，MDL在某种意义上同贝叶斯规则一样，给理论设计一套编码方案等价于要赋予一个先验概率分布。编码方案较先验概率，在某种具体条件下考虑编码方案更切实、容易一些。为了说明这一点，我们简单地介绍一下MDL是怎样应用在聚类方法中的，而不进入编码描述。

聚类似乎很难评估，分类或关联学习都有一个客观的成功判定标准，即对测试数据的预测是正确还是错误，而聚类却非如此。唯一可行的评估方法似乎是要看学习的结果，即聚类

结果是否有助于实际应用环境。（值得一提的是，这点适用于评估所有的学习方法，并非单指聚类方法。）

除了这点，聚类可以从描述长度的观点来评估。假设一种聚类学习技术将训练集E分割为k个聚类，如果这些类是天然的，用它们对E进行编码可能会是很有效的，最好的聚类方法将支持最有效的编码。

按给定的聚类方案，对训练集E中的实例进行编码的一种方法是从聚类中心（聚类中所有实例的每个属性的平均值）的编码开始，然后将E中的每个实例依据它的属性值和聚类中心的 183
关系，可能使用每个属性值与中心点属性值的差值，传送到它应属于的那一类中（用$\log_2 k$位）。使用平均值或差值，这种描述方法的前提条件需是数值属性，同时也提出了一个棘手的问题那便是怎样有效地对数值进行编码。定性属性可以用类似的方法处理，每个聚类都存在一个属性值概率分布，不同的聚类属性概率分布不同。这样，编码问题就变得很直接了：将属性值根据相应的属性概率分布进行编码，这是一种标准的数据压缩操作。

如果数据呈现出相当明显的聚类，使用这种技术将使描述长度较不用任何聚类而简单地传输训练集E中的数据要短。但是，如果聚类效果不很明显，描述长度非但不能减少，很可能要增加。这时，传输属性值的特定聚类分布所用的耗费大于对每个训练实例按相应的聚类进行编码所获的益处。这正是需要应用精密编码技术的地方。一旦聚类中心能够沟通，就有可能通过和相关的实例合作来实现自适应地传输特定聚类的概率分布：实例本身能帮助定义概率分布，概率分布又帮助定义实例。这里我们不再深入编码技术，重点是MDL公式，如被合适地运用，足以灵活地支持聚类的评估，但在实践中要得到满意的效果并不容易。

5.11 补充读物

统计学的置信度试验在大多数统计学教材中都有提到，此外还提供正态分布和学生氏分布表。（我们参考了一本非常优秀的教材，Wild和Seber （1995年）。如果你能有机会得到，我们极力推荐此书。）“Student”是统计学家William Gosset的笔名，他于1899年在爱尔兰都柏林的Guinness 酿酒厂任化学家，发明了t测试，用以处理酿酒业中使用少量的样品测试来实行质量控制。纠正重复取样t测试是由Nadeau和Bengio（2003年）提议的。交叉验证法是一种标准的统计技术，已广泛应用于机器学习中，并被Kohavi（1995年）拿来与自引导法（bootstrap）进行比较。自引导技术在Efron 和 Tibshirani（1993年）中进行了详细的介绍。

Kappa 统计量是由Cohen（1960年） 提出的。Ting（2002年） 研究了将5.7节中给出的使二类问题学习方案具有成本敏感性的算法，采用启发式的方法推广到多类问题的情形。Berry 和 Linoff（1997年） 叙述了上升图，Egan（1975年） 则介绍了使用ROC对信号检测理论的分 184
析，这项工作后又被Swets（1988年） 延续为对诊断系统的可视化和行为分析，还被应用于药物学上（Beck和Schultz 1986年）。Provost和Fawcett（1997年）带来的ROC曲线的分析方法引起了机器学习及数据挖掘界的注意。Witten等（1999年）解释了反馈率-精确率曲线在信息检索系统中的应用，van Rijsbergen（1979年）描述了F测量。Drummond和Holte（2000年）介绍了成本曲线并对它们的特性做了研究。

最短描述长度原理是由Rissanen（1985年） 陈述的，Koestler(1964年）细述了开普勒的三个行星运动规律的发现以及开普勒本人对此发现的怀疑。

Li和Vityani（1992年）引用了Asmis（1984年），提出了伊壁鸠鲁的多种解释原理。 185

第6章　实现：真正的机器学习方案

我们已经了解了机器学习方法的几种基本思路，并详细学习了怎样评估它们应用在实际数据挖掘问题中的性能表现。这样，我们做好了充分准备来了解真实的、具有产业价值的机器学习算法。目标是既要在概念上，又要在相当多的技术细节上解释这些算法，使大家能彻底理解算法并意识到实现算法过程中出现的关键问题。

事实上，第4章中所描述的最简单的方法和广泛应用于实践中的真实算法之间是有很大差距的。虽然原理相同，输入和输出——知识表达方法亦相同，但是实践中的算法更为复杂。主要是因为这些算法必须能成熟、明智地处理真实世界中的问题，如数值属性、残缺值以及最富挑战性的干扰数据。为了解各种不同方案是如何来处理干扰问题的，我们要利用在第5章中学到的一些统计学知识。

第4章以如何推理出基本规则为开场白，接着分析统计模型和决策树。然后转向规则归纳，
187 继而是关联规则、线性模型、基于实例学习的最近邻方法和聚类。除了已经在前面充分探讨过的关联规则外，本章将详述所有这些主题。

本章从决策树归纳开始，继而对C4.5系统进行完整描述。C4.5系统是素有里程碑之称的一种决策树程序，也许是到目前为止在实践中应用最为广泛的机器学习工具。接下来讨论决策规则归纳。虽然思想方法简单，但在实践中要归纳决策规则以达到能与高水准的决策树相类似的性能还是相当困难的。多数高性能的规则归纳器先找出一个初始的规则集，然后对规则集进行精练，精练则是通过一个相当复杂的优化过程，丢弃或调整个体规则使规则集能更好地工作。本章还阐述了应用于干扰数据的规则学习所基于的思想，然后介绍最近研发的使用局部决策树（partial decision tree）的方案，这个方案已被证明能达到与其他高水准的规则学习器同样好的性能，同时又避免去使用一些复杂而特别的方式。此后，再简要地来看一下怎样建立第3.5节中介绍的包含例外的规则。

支持向量机（support vector machine）的引入引起了人们对线性模型的兴趣。支持向量机是一种线性模型和基于实例学习模型的混合体，它从每个类中挑选了很少数量的、称为支持向量的临界实例，然后建立一个线性判别函数（linear discriminant function），尽可能地将各个类别分隔开。这些系统可通过在函数中包含另外一部分非线性项，从而超越了线性边界的限制，使形成二次、三次，甚至更高次的决策边界成为可能。在第4.6节中介绍的感知器可采用同样的技术来实现复杂的决策边界。一项用于扩展感知器的老技术就是将各个单元连接起来形成多层“神经网络”。所有这些思想都在6.3节中进行了叙述。

下一节的内容是阐述基于实例的学习器，展示了在第4.7节中介绍的简单的最近邻方法，并且展示了几种具有推广性的更为有效的方法。此后把适用于数值预测的线性回归法延伸为一个更为复杂的过程，从而形成第3.7节中介绍的树表达，继续讨论局部加权回归，一种用于数值预测、基于实例的策略。随后再回到聚类方法，回顾一些比简单的k均值更为复杂的方法，这些方法产生分级聚类和含有概率的聚类。最后讨论贝叶斯网络，一种非常有潜力的朴素贝
188 叶斯法的延伸，它能处理存在内部依赖关系的数据集，使朴素贝叶斯法不再那么“朴素”。

由于本章所涉及的内容特性，它与本书的其他章节有所不同。每节都可单独阅读，是独立的，在每节最后的讨论部分包含各自的补充读物。

6.1 决策树

第一个要详细演示的机器学习方案来自第4.3节所描述的、用于建立决策树的简单的分治算法。在它用于解决实际问题之前，需要在几个方面得到扩展。首先要考虑如何处理数值属性，还要考虑处理残缺值的问题。然后，还需要处理所有有关修剪决策树的重要问题，虽说用分治算法建立的决策树在训练集上表现良好，却常常由于和训练数据过于拟合而不能很好地推广到独立的训练集上。其次要考虑如何将决策树转化为分类规则。所有这些方面都受到决策树算法C4.5的引导，它和它的商业后继版C5.0已经成为用于机器学习的现成的得力工具。最后将看到C4.5和C5.0本身提供的选择方案。

6.1.1 数值属性

以前描述过的方法只是在所有属性都是名词性属性时才有效，但正如我们所了解的，大多数真正的数据集都包含一些数值属性。将算法扩展来对付这些属性并不是太难。对于数值属性，将可能性限定为双向分裂或二元分裂。假设我们使用有一些数值属性的天气数据（表1-3），那么，在考虑温度属性的第一个分裂时，牵涉到的温度值是：

64	65	68	69	70	71	72	75	80	81	83	85
yes	no	yes	yes	yes	no	no yes	yes yes	no	yes	yes	no

（重复的数值叠在一起），共有11个可能的断点，如果断点不允许将同属一个类的项目分开的话，断点就是8个。每个信息增量可以用通常的方法计算。例如，用 temperature < 71.5进行测试，产生4个yes和2个no，用 temperature >71.5 进行测试，则产生5个yes和3个no，所以这个测试的信息值是：

info ([4,2],[5,3]) = (6/14) × info ([4,2]) + (8/14) × info （[5,3]) = 0.939 bits 189

虽说采用更为复杂的方法可能会获得更多的信息，但在数值间一半的地方设定数字阈值作为概念划分界限是很普通的做法。例如，下面将看到一种最简单的基于实例学习法，它将概念划分界限置于概念空间的中间，人们还建议采用其他一些不只是包括两个最近邻样本的办法。

当用分治法建立决策树时，一旦选定第一个要被分裂的属性，就在树的顶层创建一个节点对这个属性进行分裂，算法递归地在每个子节点上继续下去。对于每一个数值属性，从表面上看，似乎必须对每个子节点的实例子集重新按照属性值进行排序，事实上这是决策树归纳程序通常使用的编写方法。但实际上并没有必要重新排序，因为在父节点的排序可以用作为每一个子节点的排序，形成一种快速的实现方法。考虑天气数据中的温度属性，它的属性值排序（这次包含重复的值）是：

64	65	68	69	70	71	72	72	75	75	80	81	83	85
7	*6*	*5*	*9*	*4*	*14*	*8*	*12*	*10*	*11*	*2*	*13*	*3*	*1*

每个温度值下面的斜体数字是拥有这个温度值的实例的序号，因此实例7的温度值为64，实例6的温度值为65，依此类推。假设决定在顶层对属性阴晴进行分裂。考虑满足阴晴= sunny的子节点。实际拥有这个阴晴属性值的实例是第1、2、8、9和11。如果斜体的顺序和样本集一

起储存（要为每一个数值属性储存一个不同的顺序），也就是说，实例7指向实例6，实例6 指向实例5，实例5指向实例9，依此类推，那么要依次读取阴晴=sunny 的实例就是一件简单的事情了。所需要的只是按照所示的次序对实例进行扫描，检查每个实例的阴晴属性值并对拥有适当属性值的实例进行记录：

9　8　11　2　1

这样，根据每个数值属性，将每个实例子集和实例的顺序一起储存，可以避免重复排序。必须在一开始就为每一个数值属性决定排列次序；以后就不再需要排序了。

当一个决策树像4.3节所述的那样对一个名词性属性进行测试时，要为这个属性的每一个可能的属性值创建一个分支。然而，数值属性的分裂被限制为二元的，这便造成了数值属性
190 和名词性属性的一个重大不同：为某个名词性属性建立完分支后，便用尽了属性所提供的所有信息，而在一个数值属性上的后续分裂还会产生新的信息。一个名词性属性在从树根到叶节点的路径中只能被测试一次，而一个数值属性则能被测试许多次。由于对单个数值属性的测试不是集中在一起而是分散于路径上的，这便造成树的凌乱而难以理解。另有一种方法，比较难以实施但可以创建较易读取的树，就是允许对数值属性进行多重测试，对树中的单独节点上进行几个不同常量的测试。一个更为简单但功效欠佳的解决方法是如7.2节将要讨论的对属性进行事先离散。

6.1.2 残缺值

对决策树建立算法的下一个改善是关于残缺值问题的处理。残缺值是现实生活中的数据集所无法避免的。正如第2章（2.4节）所解释的，一种办法是把它们当作属性的另一个可能值来处理；当属性在某些情况下缺失严重的话，这种办法是适当的。那样的话不需要再采取进一步的处理。但如果某个实例缺少属性值不特别明显的话，就需要一个更精细的解决方法。简单地忽略所有含有残缺值的实例当然是很诱人的办法，但这种解决办法过于苛刻而不太可行。有残缺值的实例往往提供了相当多的信息。有时，含有残缺值的属性在决策中并不起作用，因此这些实例和其他实例一样好。

有一个问题就是，当有一些要测试的属性有残缺值时，如何将决策树应用到这个实例中。在3.2节中示范了一个解决方法，想象将这个实例分成几个部分，采用数值加权方案，将各个部分按比例沉降在各个分支上，这个比例是指沉降在各个分支上的训练实例数量比例。最终实例的各个部分都会到达某一个叶节点，这些叶节点的决策也必须重新结合叶节点事先定好的权重而做出。在4.3节中描述的信息增益和增益率的计算同样可以应用于部分实例。使用权值代替整数累计，来计算这两个增益值。

另外一个问题就是，一旦分裂属性选定后，应该如何分裂训练集，从而在每个子节点上递归运行决策树形成过程。需要运用同样的加权程序。相关的属性值出现残缺的实例从概念上分
191 裂成几个部分，每个分支含一个部分，分裂比例就是沉降到各个分支上的已知实例的比例。这个实例的各个部分在下层节点为决策做贡献，同样可用一般的信息增益计算方法，只是它们是加了权值的。当然，如果其他属性的值也有残缺，在下层节点上可能还需要进一步分裂。

6.1.3 修剪

在第1章的劳资协商问题中，我们发现图1-3a的简单决策树实际上比图1-3b中更为复杂的

树的性能表现更好，也比较能理解。现在就来学习如何修剪决策树。

先建立一个完整的决策树然后对其修剪，采取后修剪（postpruning）策略（有时称为反向修剪）而不是前修剪（prepruning）（或称正向修剪）策略。前修剪需要在建立树的过程中决定何时停止建立子树，非常吸引人的一面，因为这能避免建立某些子树所需的全部工作，而这些子树是将来要被舍弃的。当然，后修剪确实也能提供一些优势。例如，两个属性单独不能有所贡献，但两者结合起来预测力却很强，一种两个属性的正确结合能提供很强的信息而两个属性单独运用则不能的、所谓暗码锁（combination-lock）效果。大多数决策树采用后修剪；是否能采用前修剪策略取得同样好的效果尚待进一步探讨。

在后修剪过程中考虑两种完全不同的操作：子树置换（subtree replacement）和子树上升（subtree raising）。在每一个节点上，一个学习方案也许要决定是采取子树置换还是子树上升，或者让子树和原先一样，不进行修剪。首先来观察子树置换，它是一个主要的修剪操作。它的想法是选择一些子树并用单个叶节点来代替它们。例如，图1-3a中包括2个内部节点和4个叶节点的整个子树，被单个叶节点bad所替换。如果原先的决策树是由前面所述的决策树算法建立的，这当然会引起在训练集上的正确率下降，因为算法要继续创建决策树直至所有的叶节点都是纯粹的（或者说直至所有的属性都被测试过）。不过这也许会提高在独立选出的测试集上的正确率。

当实施子树置换后，它是从叶节点向树根方向进行处理的。在图1-3的例子中，整个子树不会马上被置换成如图1-3a。首先，要考虑将健康计划子树上的3个子节点替换成单个叶节点。假设做出要进行更换的决定，马上会讨论如何做出这个决定。然后，继续从叶节点开始工作， 192
考虑将现在只有两个子节点的每周工作时间子树换成单个叶节点。在图1-3例子中，这个替换实际上也已完成，图1-3a中的整个子树被替换成了标为bad的单个叶节点。最后，考虑用单个叶节点来置换第1年工资增长子树的两个子节点。在这个例子中这个决定未能实现，因此决策树保持如图1-3a所示。后面将要讨论这些决定实际上是如何做出的。

第二种修剪操作子树上升，更复杂，是否有必要也不是很清楚。在这里描述它，是因为它被应用于有影响力的C4.5决策树系统。子树上升在图1-3例子中没有发生，所以用图6-1所示的虚拟例子来解释。这里，考虑对图6-1a中所示的树进行修剪，修剪结果显示在图6-1b中。整个自C以下的子树被提升上来置换子树B。注意虽说这里的B和C的子节点是叶节点，但它们也可以是完整的子树。当然，如果要进行上升操作，有必要将标有4和5的节点处的样本重新划分到标为C的新子树中。这就是为何那个节点的子节点被标为：1'、2' 和3' ——表明它们并不与原来的子节点1、2和3相同，而是包含了原本被4和5涵盖的样本。

子树上升可能是一个耗时的操作。在实际的实现中，它被限制在只能提升最为普及的分支。就是说，由于从B至C的分支较之从B至节点4或者B至节点5的分支有更多的训练样本，才考虑做图6-1中的上升。否则，如果（举个例子）节点4是B的主要子节点，将考虑上升节点4去代替B，并重新对所有C以下的样本以及节点5的样本进行分类以加入到新的节点。

6.1.4 估计误差率

两种修剪操作就讨论到此。现在必须来解决怎样决定是否要用单个叶节点来替换一个内部节点（子树置换），或者是否要用位于一个内部节点下面的某个节点来替换它（子树上升）的问题。为了能做出理性的决定，必须用一个独立的测试集在某个节点处进行期望误差率估

计。既要在内部节点处又要在叶节点处进行误差估计。如果有了这样的估计值，只要简单地比较子树的估计误差和其要替换成的子树的估计误差，便能明确地决定是否要对某个子树进行置换或上升操作。在对预备提升的子树进行误差估计之前，当前节点的同胞节点所含的样
193 本，即图6-1中节点4和节点5所含样本必须暂时重新分类到被提升的树中。

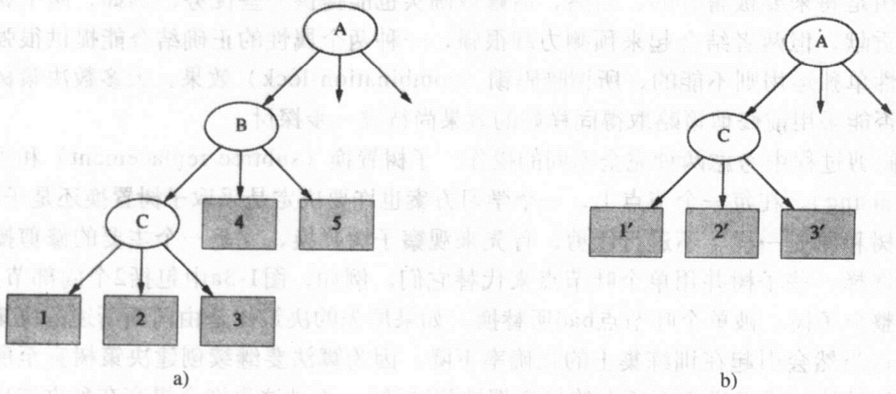


图6-1　子树上升例子，节点C"上升"并包含节点B

用在训练集上的误差作为误差估计是无效的，将导致不会有任何修剪，因为树是特为训练集建立的。一种取得误差估计的办法是采用标准验证技术，保留部分原始数据作为一个独立的测试集来估计每个节点上的误差。这称为减少－误差（reduced-error）修剪。它的缺点是决策树是在较少的数据上建立。

另一个方法是试图以训练集本身为基础来估计误差。这正是C4.5算法中所做的，我们在这里介绍一下这个方法。这是一种基于一些统计推理的方法，但这个统计基础有点薄弱并且比较特殊。然而，在实际运用中似乎效果良好。它的想法是考虑到达每个节点的实例集，并想象选择多数类来代表这个节点。这便能提供一个在实例总数N中所占的"错误"数量E。现在想象在节点上的错误的真实可能性是q，并且N个样本是由参数为q的伯努利程式所产生的，E成为了错误数量。

这个情形如同在5.2节中讨论旁置法时所考虑的情形，已知某个观察到的成功率，计算真实成功率p的置信区间。有两点不同。一个是微不足道的：这里讨论的是误差率q而非成功率p；它们之间存在简单的关系$p + q = 1$。第二个比较重要：这里数据E和N来自训练集，而在5.2节中考虑的是使用独立的测试集。由于这个不同，需要设定置信度上限来为误差率做一个
194 较为悲观的估计，而不是给出估计值置信区间。

这里用到的数学知识是和前面一样的。给定某个特定的置信度c（C4.5所使用的默认值c =25%），找出它的置信边界z，使得

$$\Pr\left[\frac{f-q}{\sqrt{q(1-q)/N}} > z\right] = c$$

这里N是实例的数量，$f = E / N$是观察到的误差率，q是真实误差率。和前面一样，这将得到q的一个置信度上限。现在就用这个置信度上限为在节点上的误差率e做一个（悲观的）估计：

$$e=\frac{f+\frac{z^2}{2N}+z\sqrt{\frac{f}{N}-\frac{f^2}{N}+\frac{z^2}{4N^2}}}{1+\frac{z^2}{N}}$$

注意在分子的平方根前面用 + 号来获得置信度上限。这里，z是对应于置信度c的标准差，当c =25% 时，z =0.69。

要知道在实际中怎样计算，再来看一下图1-3的劳资协商决策树，图6-2对其中特别的部分重新展示添加了到达叶节点的训练实例的数量。利用前面的公式，置信度设为25%，那么z=0.69。考虑底层左边的叶节点，E = 2, N = 6, 因此f = 0.33。将这些数据代入公式，误差置信度上限计算出为e = 0.47。这意味着将要用47%这个悲观的误差估计来替代在这个节点训练集上得到的33%的误差率。这确实是悲观的估计，对于一个二类问题，误差率若高于50%便是很大的失误。与这个节点的相邻叶节点E = 1, N = 2, 由于计算出误差置信度上限为e = 0.72，情况就显得更糟糕了。第三个叶节点e值和第一个是相同的。下一步是将这三个叶节点的误差估计根据它们各自所覆盖的实例数量的比率6:2:6，进行组合，得到组合误差估计值0.51。现在来考虑它们的父节点健康计划。这个节点覆盖了9个类值为bad的实例和5个类值为good的实例，因此在训练集上的误差率是 f = 5/14。根据这些数据，按照前面的公式计算得到一个悲观的误差估计e = 0.46。由于这个值小于三个子节点的组合误差估计，因此子节点便被修剪掉了。

下一步处理每周工作时间节点，它现在含有两个都为叶子的子节点。第一个子节点E = 1，N= 2，它的误差估计为e = 0.72，第二个正是先前讨论的e = 0.46。把它们按照比率2:14组合起来，得到一个比每周工作时间节点的误差估计高的值，因此子树也被修剪掉，用一个叶节点来代替。 195

从这些样本中获得的误差估计数字需要有所保留，因为这种估计是一种启发式估计并且是建立在几个虚弱的假设上：置信度上限的使用、正态分布假设以及采用在训练集上取得的统计数据这个事实。但是不管怎样，误差公式在量化上是正确的，而且这个方法似乎在实践运用中效果良好。如有必要，所应用的置信度水平25%，可以修正一下以取得更满意的结果。

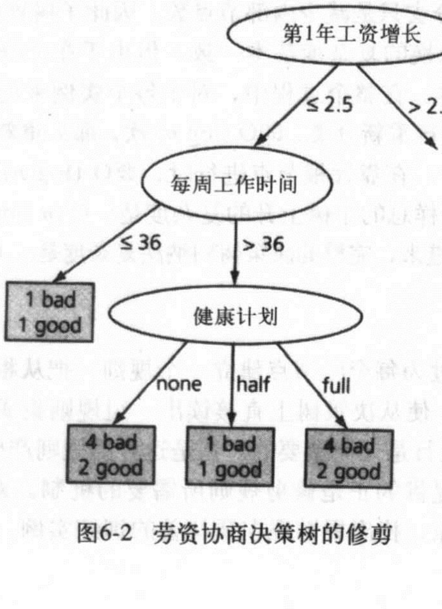


图6-2 劳资协商决策树的修剪

6.1.5 决策树归纳的复杂度

已经学习了如何完成修剪操作，了解了决策树归纳法的所有中心部分。现在来思考决策树归纳法的计算复杂度。使用标准符号：O(n)，表示一个随n线性增长的数量，O(n^2)是随n的二次方增长的量，诸如此类。

假设训练集拥有n个实例以及m个属性。我们需要对树的尺寸做一个假设，假设树的深度是由log n决定的，即O(log n)。这是一个含有n个叶节点的树的标准增长率，假设树保持“浓
196 密”生长，没有退化成非常深且长丝状的分支。注意这里假设大多数的实例都是不同的，也就相当于m个属性能提供足够多的测试，使实例各不相同。例如，如果只存在少数几个二类属性，那么它们只允许有这么多的不同实例，树的生长不可能超出某个限度，表现这种“在限定范围内”的分析是毫无意义的。

建树的计算成本首先是

O (mn log n)

考虑一个属性在树的所有节点上所要做的工作量。当然不必在每一个节点上考虑所有的实例。但在树的每一层，必须考虑含有n个实例的整个实例集。由于树有log n个不同的层，处理一个属性所需要的工作量便是O (n log n)。在每个节点上所有属性都要被考虑，因此总的工作量是O (mn log n)。

这个推理有几个前提假设。如果一些属性是数值型的，它们必须是经排序的，一旦最初的排序完成后，如运用了适当的算法（6.1节），就没有必要在树的每一层进行再次排序。最初的排序操作对于每个属性工作量为O (n log n)，共有m个属性：因此前面的复杂度数值没有变化。如果属性是名词性的，没有必要在每个节点上考虑所有属性，因为在树的上部已用过的属性就不能再用了。然而，如果是数值属性，属性可以再次应用，因此必须在树的每一层考虑所有的属性。

下一步考虑子树置换修剪操作。首先对于每个树节点都要做一个误差估计。假设使用了适当的累计操作，它便是与树所含的节点数存在线性关系。对每个节点都要做置换考虑。树最多含有n个叶节点，每个节点含一个实例。如果是二分树，每个属性都是数值型的或是二值属性，那么便有$2n-1$个节点；多路分支只是减少内部节点数。因此子树置换操作复杂度是O (n)。

最后，子树上升和子树置换的复杂度基本一致。但由于在上升操作中需要对实例进行重新分类而产生了一个额外成本。在整个过程中，对于每个实例来说，也许需要在它的叶节点和根节点之间的每个节点处进行重新分类，即O (log n) 次。那么重新分类总次数为O (n log n)。
197 并且重新分类并非是单个操作：在靠近根节点进行时，要O (log n) 次操作，在平均高度处进行时则需要这个数的一半。这样总的子树上升的复杂度是：O (n (log n)^2)。

这样，将所有的操作考虑进来，完整的决策树归纳法复杂度是：O (mn log n) +O (n(log n)^2)。

6.1.6 从决策树到规则

正如3.3节中提到的，通过为每个叶节点建立一个规则，把从根节点到叶节点的路径中遇到的所有测试条件联合起来，使从决策树上直接读出一组规则集成为可能。这样创立的规则是清楚明确的，以什么顺序执行是无关紧要的。但是这样的规则产生某些不必要的复杂。

上面讲到的误差率估计提供的正是修剪规则所需要的机制。对某一个具体的规则来说，考虑将它的每个条件暂时去除，找出规则现在所涵盖的训练实例，用这些实例计算出新规则

的误差率的悲观估计，并将此与原来规则的误差率的悲观估计相比较。如果新规则的结果比较好，则删除这个条件，然后继续找其他条件进行删除。当删除任何条件都不再能改善规则时，则保留规则。一旦所有的规则都用这种方法修剪过了，有必要看一下是否有重复的规则，应把它们从规则集里去除。

这是一个用于探测规则中重复条件的贪心方法，不能保证最好的条件组合是否会被删除。一种改进是考虑所有的条件子集，但是这样的代价太高了。另外一种解决办法是采用优化技术，诸如模拟退火或者遗传算法来选择这个规则的最佳版本。但不管怎么说，这个简单的贪心方法似乎可以产生相当好的规则集。

即使采用贪心方法，计算成本也是一个问题。对于每个条件，它们都是被删除对象的候选者，规则的效果必须在所有的训练实例上进行重新评估。这意味着从树中形成规则会是很慢的过程。下一节要介绍一种较快的方法，它是直接产生分类规则而不用先形成决策树。

6.1.7 C4.5: 选择和选项

我们以讨论标志性的决策树方案C4.5以及其后继者C5.0在实践中的一些应用来结束决策树的学习。这些是由J. Ross Quinlan从20世纪70年代末期开始，用了20年的时间来设计的。20世纪90年代初期，关于完整C4.5描述是一本优秀的可读性很强的书（Quinlan，1993年），附有全部源代码。更新的版本C5.0已经商业化。它的决策树归纳基本和C4.5相同，测试结果略 198
有不同，但改进不明显。虽然在公开文献上未见报道，但它的规则产生迅速并显然运用了不同的技术。

C4.5本质上就如上节中描述的。缺省置信值设定在25% 并在绝大多数场合运行良好；如果在测试集上，决策树的实际误差率比估计的高出许多的话，也许这个置信度可以设得更低一些，从而导致更"猛烈"的修剪。另外还有一个重要参数，它的作用是消除那些几乎在所有训练实例上结果都相同的测试。这类测试经常是没有什么用的。因此，测试不会加入到决策树中，除非它们至少能产生两个结果，即覆盖实例数量达某个最小值。这个最小值的缺省值是2，但这个数是可以调控的，对于含许多干扰数据的情况也许应提高这个值。

6.1.8 讨论

自上而下的决策树归纳法或许是数据挖掘中应用最广的机器学习研究方法。研究者们对学习过程中几乎所有可能的方面都进行了各种探究，例如，不同的属性选择标准或修改了的修剪方法。可是很少能在不同数据集上使正确率得到真正的改善。有时使用不同的修剪方法，归纳树尺寸是明显减小了，但是将C4.5的修剪参数设小一点经常也能获得同样的效果。

在关于决策树的讨论中，我们假设在每一个节点上，仅仅用一个属性来将数据分裂成子集。然而，也可以允许测试条件一次牵涉到几个属性。例如，对于数值属性，每个测试可以是属性值的线性组合。那么最终形成的树是如4.6节中讨论的分级结构线性模型组成，分裂也不再局限于轴平行。相对于常用的单变量（univariate）树，测试牵涉到一个以上属性的树称为多元（multivariate）决策树。多元测试是由用于决策树学习的分类和回归树（CART）系统引入的（Breiman等，1984年）。它们通常更精确，比单变量树更小，但生成树的时间要长一些并且较难解释。在7.3节中将简要地介绍一种使用主分量分析（principal component analysis）法来产生它们的方法。 199

6.2 分类规则

为生成规则所用的基本覆盖算法在4.4节中称为是一种割治（separate-and-conquer）技术，它先确定一条规则能覆盖属于某一个类的实例（并排除不属于此类的实例），将这些实例分离出来，然后继续对所剩的实例进行处理。这种算法被当作是许多生成规则系统的基础。我们还描述了一个简单的基于纠正（correctness-based）的方法用于选择在每个阶段对规则增加何种测试。但是，还有许多其他可能方法，具体方案的选择对规则生成具有重大影响。在这一节中要考察为了选择测试所采用的不同方案。还要讨论怎样通过处理残缺值和数值属性将基本的规则生成算法应用到实践中。

所有这些规则生成方案的真正问题在于它们有对训练数据过度拟合的倾向，不能很好地推广到独立的测试数据集上，特别是在干扰数据上。为了能生成适合干扰数据的好规则，有必要使用某些方法来衡量单个规则的真正价值。评估规则价值的标准方法就是评估规则在某个独立实例集上的误差率，这个独立实例集是从训练集中保留出来的一部分，我们将在后面解释这个问题。然后讨论两种具有产业价值的规则学习器：一种是将简单的割治技术和全局优化（global optimization）步骤结合起来的方法，另一种是重复建立局部决策树(partial decision tree)并从中提炼出规则。最后讨论怎样产生含有例外的规则，以及例外的例外。

6.2.1 选择测试的标准

在4.4节中介绍基本规则学习器时曾指出必须找出一种方法，从许多可能的测试中选定某个测试加到规则中，以避免规则覆盖任何否定实例。为达此目的，使用能使p/t比率达到最大值的测试，这里t是指新规则所覆盖的样本总数，p是其中肯定的数目，即指属于问题中的类。试图使规则的“正确性”最大化，所覆盖的肯定样本比率越高，规则就越是正确。另一种方法是计算信息增益（information gain）

$$p\left[\log\frac{p}{t}-\log\frac{P}{T}\right]$$

和前面一样，p和t分别是新规则所覆盖的肯定实例数量和所有实例数量，P和T分别是添加新
200 测试之前、规则所覆盖的对应的实例数量。原理是它代表了关于当前肯定样本的总信息增益，是由满足新测试的样本数量乘以信息增益所给出。

选择加入到规则中的测试的基本标准是找出尽可能多地覆盖肯定样本、并尽可能少地覆盖否定样本的测试。原始的基于正确性的方法，只是看肯定样本在规则所覆盖的所有样本中所占的百分比，而不管规则究竟覆盖多少数量的肯定样本，当它没有覆盖任何否定样本时便会得到一个最大值。因此使规则精确的测试便优先于使规则不精确的测试，不管前者所覆盖的肯定样本数有多么小，也不管后者所覆盖的肯定样本数有多么大。例如，如果有一个候选的测试覆盖1个样本，这个样本是肯定样本，那么这种标准将使这个测试优先于另一个能覆盖1 000个样本其中含1个否定样本的测试。

另一方面，基于信息的方法更注重于规则所覆盖的大量肯定样本而不管所建的规则是否精确。当然，两种算法都是持续增加测试直到最终的规则是精确的为止，这意味着使用基于正确性的方法将会较早结束，而使用基于信息的方法将会增加更多的测试项。因此基于正确性的方法可能会发现一些特殊情形，完全排除它们，避免了后面规则的大量描绘（由于特殊情形已经处理了，后面更具推广性的规则也许更为简单）。基于信息的方法先要试图产生高覆

盖的规则而将特殊情形放在后面处理。使用两种策略产生精确的规则集，哪种更有优势并不明显。而且，事实上正如下面将要讲到的，规则集可能要被修剪并且要容许不精确的规则存在，因此整个情形变得更为复杂。

6.2.2 残缺值，数值属性

如同分治决策树算法一样，实践中令人厌烦的对于残缺值和数值属性的考虑是无法回避的问题。事实上，也没有什么可以多说的。现在我们已知道在决策树归纳中是如何解决这些问题的，那么适用于规则归纳的解决方案便很容易得到了。

当使用覆盖算法产生规则时，处理残缺值的最好方法就是把它们当作不符合任何测试。这在产生决策列时特别适当，因为它会促进学习算法利用肯定会成功的测试来将肯定实例分离出来。这样的效果是要么使用不牵涉到残缺值属性的规则来对残缺值实例进行处理，要么 201
在绝大多数实例都已经处理完成后再来处理这些实例，这时测试可能涵盖的是其他属性。覆盖算法应用于决策列相对于决策树来说，在这方面有一个明显的优势：有麻烦的样本可以稍后处理，这样由于绝大多数样本都已经分类完成并从实例集中去除，那时它们就显得不那么麻烦了。

数值属性可以采用它们在决策树中的相同办法处理。对于每一个数值属性，根据属性值将实例进行排序，对于每一个可能的阈值，考虑一个二分的小于/大于测试，评估则完全采用对二值属性进行评估的方法。

6.2.3 生成好的规则

假设你并非要生成对所有训练集实例都能正确分类的完美规则，而是要生成“敏感”规则，能避免对训练集实例过度拟合从而对于新的测试实例能有较好的性能表现。怎样决定哪些规则是有价值的呢？为了排除个别讨厌的错误实例，要持续在规则中添加测试项，但同时也会排除越来越多的正确实例。怎样知道何时开始起反作用了呢？

下面来看关于表1-1所列隐形眼镜问题的几个可能的规则，有些是好的，有些是坏的样本。首先来看规则

```
If astigmatism = yes and tear production rate = normal
  then recommendation = hard
```

这条规则覆盖6种情形，其中4种能得到的是正确结论；因此它的成功因数就是4/6。假设又添加了一个测试项使之称为“完美”规则：

```
If astigmatism = yes and tear production rate = normal
  and age = young then recommendation = hard
```

这使正确率提高为2/2。哪条规则更好呢？第二条规则对训练集数据具有较高的正确率，但是只能涵盖2种情形，而第一条规则却能涵盖6种情形。第二种规则或许是对训练数据过度拟合了。在规则学习实践中，需要有一种基本方法能用于选择较合适的规则版本，即选出能在未来的测试数据上获得最高正确率的规则。

假设将训练数据分裂成两个部分，称为成长集（growing set）和修剪集（pruning set）。成长集是用于生成规则的，这个规则生成是运用基本的覆盖算法。然后从规则中去除某个测试项，并使用修剪集来对这个经修剪的规则进行结果评估，看看是否比原先的规则更好。重复这样的修剪过程直至去除任何测试项都不再能使规则有任何改进。对每个类都重复整个过 202

程，为每个类获得一条最好的规则，而总体最好规则是通过使用修剪集对这些规则进行评估而获得的。将这条（最好）规则添加到规则集中，将它所覆盖的实例从训练数据中去除，成长集和修剪集中都要去除，并重复整个过程。

为何不在建立规则时进行修剪而是在建完之后才丢弃其中的某部分呢？即为何不采用先修剪而采用后修剪呢？就像决策树修剪时那样，最好是先生成一个最大的树，然后再往回修剪，对于规则也是同样的，最好是先形成最完美的规则然后再对它进行修剪。谁知道呢？最后一个测试项的添加也许会形成一条相当好的规则，这也许是在采用大胆的先修剪策略时绝不会注意到。

成长集和修剪集必须要分隔开是很关键的，因为使用形成规则的数据来评估一条规则是会产生误导的：会由于过度拟合的规则获得优先而导致严重错误。通常2/3的训练数据用于成长，而1/3的训练数据用于修剪。缺点是算法只在成长集数据上进行学习，若某些关键实例被分隔在修剪集中，算法可能会遗失一些重要规则。另外，错误的规则有可能获得优先，因为修剪集只含1/3的训练数据也许并不能完全具有代表性。这些影响可以通过在算法所进行的每次循环中，即在每条规则最终选择之后，将训练数据重新分裂为成长集和修剪集来改进。

使用隔离的修剪集进行修剪，它既适用于决策树也适用于规则集，称为减少-误差修剪法(reduced-error pruning)。前面描述的在规则每次一生成就立即修剪规则称为递增（incremental）减少-误差修剪。另一种可能方法是先建立未修剪的完整规则集，然后再丢弃个体测试来进行修剪。然而这种方法慢多了。

当然，有多种不同方法能以修剪集为基础，评估一条规则的价值所在。一种简单的方法是在闭合形式的假设条件下，如果某个规则是某个理论的唯一规则，考虑应用这个规则从其他类别中识别出预测类究竟完成得有多好。如果规则所覆盖的t个实例中能得到p个实例是正确的，并且在所有的T个实例中含有P个实例是属于这个类的，那么就说它能得到p个肯定实例是正确的。规则所没有覆盖的否定实例有$N-n$个，这里$n=t-p$是指规则所覆盖的否定实例数量，$N=T-P$是否定实例的总数量。因此，规则有一个总体成功率

203

$$[p+(N-n)]/T$$

在测试集上评估所得到的这个数量用来评估采用减少-误差修剪所得到的规则的成功性。

由于这种方法视未被规则覆盖的否定样本的重要性，等同于规则所覆盖的肯定样本的重要性，是不切实际的，实际情况是被评估的这条规则最终将伴随着其他规则共同工作，因此对这种方法批评意见较多。例如，一条规则所覆盖的3 000个实例中有$p=2\ 000$个实例得到正确预测（即有n=1 000个是错误的），相比另一条规则覆盖1 001个实例，其中有$p=1\ 000$个实例得到正确预测（即有n=1个是错误的），评估结果是前者比后者更成功，因为在第一种情形中$[p+(N-n)]/T$等于$[1\ 000+N]/T$，而在第二种情形中却只有$[999+N]/T$。这是违背直觉的：第一条规则预测能力明显比第二条差，因其误差率是33%对0.1%。

利用成功率p/t来衡量，就像覆盖算法的原始表达式（图4-8）那样，也不是完美的解决方法，因为它会选择能得到1个正确预测(p=1)而总覆盖数为1（因此n=0）的规则优先于更有用的、能从1 001个实例中得到1 000个正确预测的规则。另一种曾被应用的方法是计算$(p-n)/t$，但它也存在同样的问题，因为$(p-n)/t=2p/t-1$，所以在比较两条规则时，结论等同于两个成功率的比较。看来似乎很难找到一种简单的方法来衡量规则的价值，这个价值要在所有情形下都能和直觉判定相符。

无论使用何种衡量规则价值的方法，递增减少-误差修剪算法都是相同的。图6-3给出了一种以这种思想方法为基础的可能的规则学习算法。它产生一组决策列，依次为每个类生成规则，在每个阶段根据它在修剪数据上得出的价值，选择出最好的规则。应用生成规则的基本覆盖算法（图4-8）来为每个类生成好的规则，并利用先前讨论的p/t正确率量度来选择加入规则的条件。

```
将E初始化为实例集
将E 按2:1分裂为成长集和修剪集
    对在成长集和修剪集中都含有实例的每个类 C
        使用基本覆盖算法，为类别C生成最好的规则
        在修剪集上计算规则的价值w(R)，以及将最后一个条件去除后的价值w(R－)
        当w(R－)>w(R)时，从规则中去除最后一个条件并重复上一步骤
从所生成的规则中选择w(R)值最大的那个规则
输出规则
从E 中去除被这个规则所覆盖的实例
继续循环
```

图6-3 递增减少-误差修剪法形成规则的算法

这个方法用于产生规则归纳方案，这些方案能处理大量数据并且运行速度很快。它还可以提速，这是通过为排序的类生成规则，来替换原先在每个阶段为每个类生成一条规则并选择最好的。一种较合适的排序是按照各个类在训练集中出现次数的升序排列，因此最少出现的类第一个处理，最常见的类最后处理。可获明显提速的另一方面是当规则产生某个足够小的正确率时，即停止整个程序，这样便可以避免在最后为生成许多覆盖量很小的规则而花费时间。但是，非常简单的停止条件（比如，当一条规则的正确率低于它所预测的类的缺省正确率时即停止）不能获得最好的性能表现，而目前所发现的似乎能提高性能表现的条件是相当复杂的，它们是以最短描述长度原理为基础的。

6.2.4 使用全局优化

通常，如此使用递增减少-误差修剪算法来生成规则，特别是在大的数据集上，效果相当不错。但是，在规则集上使用全局优化步骤能获得有价值的性能优势。动机是通过修改或替换个体规则来提高规则集的正确率。实验显示使用后归纳（post induction）优化，规则集的大小和性能都获明显改善。但另一方面，程序本身相当复杂。

为了对这个精巧的、富有启发的以及具有产业价值的规则学习器是怎样形成的提供一个概念，图6-4展示了一种称为RIPPER的算法，它是“重复递增修剪以减少误差”的缩写。检验是按照各个类（所含实例数量——译者注）由小到大的顺序来进行，类的初始规则集是用递增减少-误差修剪算法产生的。引入一个额外的停止条件，它是依赖于样本和规则集的描述长度。描述长度DL有一个复杂的公式，它要考虑传送关于某组规则的一组样本所需的位数，传送带有k个条件的规则所需的位数，以及传送整数k所需位数之和，乘以50% 用以补偿可能发生的重复属性所需的位数。在为某个类建立了规则集之后，重新考虑每条规则，产生两种变体，再次使用减少-误差修剪法，但在这个阶段，把为这个类而建的其他规则所覆盖的实例从修剪集中去除，在剩余的实例上所获的成功率用来作为修剪标准。如果两种变体中的某一种产生一个较好的描述长度，便用它来替换规则。接下来再次激活最初的创建规则阶段，对 204～205

任何属于这个类而未被规则覆盖的实例进行扫尾工作。为了确保每条规则对减少描述长度都做了贡献，在进行下一个类的规则生成程序之前要做最后的检查。

将E初始化为实例集
对每个类C，从最小到最大
　　建立：
　　　　将E按2：1分裂为成长集和修剪集
　　　　重复循环直至（a）不存在未被覆盖的、类别为C的实例；或(b) 规则集和实例集的描述长度(DL)比目前找到的最短描述长度大64位以上；或（c）误差率超过50%：
　　　　　　成长阶段：贪心式地增加条件来建立规则，直至规则达到100%正确率，（所增加的条件是）通过测试每个属性的每种可能属性值，并选择其中能获得最大信息增益G的条件
　　　　　　修剪阶段：从最后一个条件往前依次修剪条件，只要规则的价值W上升则继续
　　优化：
　　　　产生变体：
　　　　对于类别C的每个规则R，
　　　　　　重新将E分裂为成长集和修剪集
　　　　　　从修剪集中去除类别C的其他规则所覆盖的实例
　　　　　　使用GROW和PRUNE在新分裂的数据上产生并修剪两个竞争规则
　　　　　　　　R1是一个重建的新规则
　　　　　　　　R2是通过贪心式地在R中添加测试条件来生成的
　　　　　　采用A衡量（代替W）在减少的数据上进行修剪
　　　　选择代表：
　　　　将R、R1和R2中描述长度最小的一个来代替R
　　扫尾：
　　　　如果还有未被（规则）覆盖的、属于类别C的残余实例，则返回到建立（BUILD）阶段，在这些实例上建立更多的规则
　　整理：
　　　　为整个规则集计算DL(描述长度)并依次除去规则集中的每个规则计算DL；去除使DL增加的规则
　　　　去除由生成的规则所覆盖的实例
继续循环

a) 规则学习算法

DL：描述长度

$G=p[\log(p/t)-\log(P/T)]$

$$W=\frac{p+1}{t+2}$$

$$A=\frac{p+n'}{T}$$；规则的正确率

p = 规则所覆盖的肯定样本数量（正确的肯定）
n = 规则所覆盖的否定样本数量（错误的否定）
$t = p+n$；规则所覆盖的样本总数
$n' = N-n$；未被规则覆盖的否定样本数量（正确的否定）
P = 属于这个类的肯定样本数量
N = 属于这个类的否定样本数量
$T = P+N$；属于这个类的样本总数

b) 符号的意义

图6-4　RIPPER

6.2.5 从局部决策树中获得规则

规则归纳还有另外一种方法，它避免了全局优化但能生成一个精确的、紧凑型的规则集。这个方法将决策树学习所应用的分治策略和规则学习的割治策略结合起来。它是这样采用割治策略的，先建立一条规则，将规则所覆盖的实例去除，然后递归地为剩余的实例建立规则，直至没有剩余实例。它与标准方法的不同之处在于每条规则都是创建出来的。在本质上，创建一条规则就是先为当前的实例集创建一个经修剪的决策树，将覆盖实例最多的叶节点转换成一条规则，然后丢弃这个决策树。

重复建决策树只是为了要丢弃绝大部分树，但它的前景并不像表面看起来的那样古怪。利用经修剪的树来得到一条规则，取代通过每次去除一个联合条件递增地修剪规则，可以避免过度修剪倾向，过度修剪是基本的割治规则学习法的一个特殊问题。将割治学习法和决策树结合起来能提高灵活性和速度。建立完整决策树只是为了要得到一条规则，的确是很浪费，206~207
然而这个程序还可被显著加速而不牺牲上述优点。

关键思路是建局部决策树而不展开整个树。局部决策树是一个普通的决策树，它包含了一些未定义子树的分支。为了建这样的树，将建构和修剪操作结合成一体以便能找到一个“稳定”的、不能再简化的子树。一旦找到这样的子树，建树过程即停止并从中读出一条规则来。

图6-5总结了建树的算法：它将一组实例递归地分裂成一个局部树。第一步是选择一种测试条件将实例集分成几个子集。这个测试条件的决定是采用建决策树时常用的信息增益法（4.3节）。然后将子集按它们的平均熵的升序依次展开。这样做的原因是排在后面的子集很可能不因这次展开而结束，而平均熵较低的子集更可能导致形成小的子树，从而产生一个更通用的规则。递归进行这个过程直到这个子集展开成为叶节点，然后返回继续下一步。一旦出现一个内部节点，它的所有子节点都成了叶节点，算法就立即检查是否用一个叶节点来替换这个内部节点会更好。这正是用于决策树修剪的标准的“子树置换”操作（见6.1节）。如果执行了置换操作，算法接着按标准方式返回，展开这个新近置换节点的同胞节点。但是，如果在返回时，遇到某个节点，它的所有已展开的子节点并不全是叶节点，当有一个潜在的子树置换没有执行时，就会发生剩下的子集不再展开，相应的子树也就未被定义。由于算法采用了递归结构，这种情况的出现会自动终止建树过程。

```
展开子集（S）:
    选择一个测试T，将实例集分裂为子集
    将子集按平均熵的升序排列
    当（存在一个未被展开的子集X，
        并且目前已展开的所有子集都是叶节点），则
        展开子集（X）
    如果（所有的已展开的子集都是叶节点，
        并且子树估计误差 > 节点估计误差）
        取消子集的展开并且将该节点变为一个叶节点
```

图6-5 将样本展开成局部树的算法

图6-6展示了一个逐步建局部树的例子。从图6-6a到图6-6c的各个阶段，是按一般的方式进行递归建树，除了每次展开都是从同胞节点中选择含最低熵的那个节点进行：从图6-6a到 208
图6-6b阶段是节点3。灰色椭圆节点是目前还未展开的节点；长方形节点是叶节点。从图6-6b

到图6-6c，长方形节点比同胞节点（节点5）的熵低，但由于它是叶节点已不能再展开了。这时便进行返回过程，节点5被选择进行展开。一旦到达图6-6c阶段，出现了一个节点5，它的所有子节点都被展开成为叶节点，这触发了修剪操作。考虑并接受了对节点5进行子树置换，进入图6-6d阶段。现在又要考虑对节点3进行子树置换，这个操作也被接受了。继续进行返回操作，节点4的熵比节点2小，因此被展开成2个叶节点。现在要考虑对节点4的子树置换了：假设节点4没有被置换。这时程序便在图6-6e阶段终止，形成带有3个叶节点的局部树。

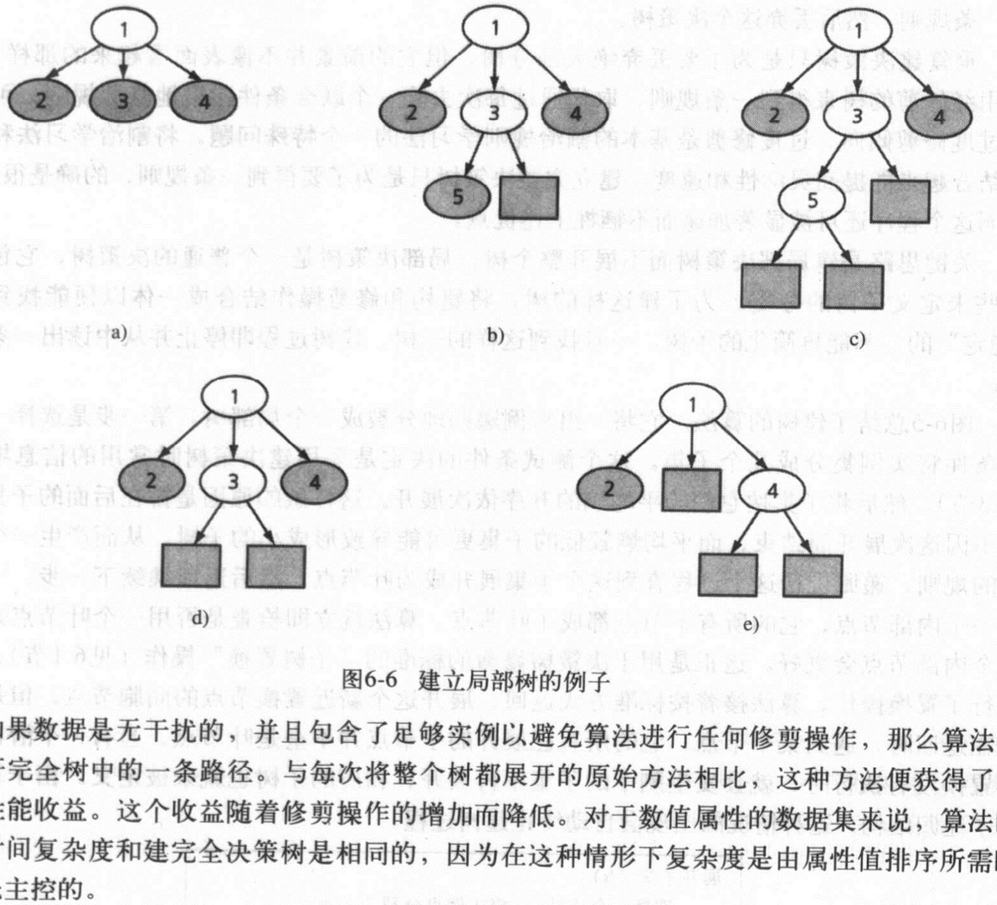


图6-6　建立局部树的例子

如果数据是无干扰的，并且包含了足够实例以避免算法进行任何修剪操作，那么算法只会展开完全树中的一条路径。与每次将整个树都展开的原始方法相比，这种方法便获得了最
209 大的性能收益。这个收益随着修剪操作的增加而降低。对于数值属性的数据集来说，算法的渐近时间复杂度和建完全决策树是相同的，因为在这种情形下复杂度是由属性值排序所需时间首先主控的。

一旦建立了局部树，便能从树中提炼出一条规则。每个叶节点对应于一条可能的规则，我们要在那些被展开成叶节点的子树（少数的几个子树）中寻找一个"最好"的叶节点。试验显明最好是选择覆盖实例数量最多的叶节点以得到最为通用的规则。

当一个数据集含有残缺值时，可以用建决策树时使用的相同方法。如果一个实例由于残缺值问题而不能分到任何分支中，就将它按照权值比例分到每个分支，权值比例是将各个分支所含的训练实例数量，用节点所含的所有已知训练实例数量进行正常化得到的。在测试中，对每条规则也使用同样的程序，这样便可将权值和每条规则在测试实例上的应用联系起来了。在进行规则列中的下一条规则测试之前，将这个权值从实例的总权值中扣除。一旦权值减到0，便将预测类的概率根据权值组合起来形成最终分类结果。

这便产生了一个用于干扰数据决策列学习的简单却出乎意料有效的方法。对比其他一些

规则形成方法，它的主要优势在于简便，因为其他方法需要采用复杂的全局优化才能达到相当的性能水准。

6.2.6 包含例外的规则

在3.5节中曾学习到规则的自然延伸便是允许它们有例外，以及例外的例外等等。实际上整个规则集可以被看作是当没有应用其他规则时的一个缺省分类规则的例外。利用前一节所述的某种衡量方案来产生一个“好”规则的方法，正是提供了一种产生包含例外的规则所需的机构。

首先，为最顶层的规则选择一个缺省类，自然是使用在训练集中最常出现的类。然后，找到关于某个类的一条规则，这个类是指任何一个有别于缺省值的类。自然是要在所有类似的规则中寻找最有辨别能力的，比如，在测试集上评估结果最好的那个。假设这条规则有如下形式

如果 <条件满足> 那么 类 = < 新的类 > 210

用它可将训练集数据分裂成两个子集：一个包含所有能满足规则条件的实例，另一个则包含所有不满足规则条件的实例。如果任何一个子集中含有属于一个以上的类的实例，便在这个子集上递归调用算法。对于能满足条件的子集来说，“缺省类”就是规则中所指定的新的类；对于不能满足条件的子集来说，缺省类就保持原来所定的类。

对于3.5节中所给出的含有例外的规则，这些规则是关于表1-4的鸢尾花数据的，下面来检验这个算法是如何工作的。我们用图6-7所示的图形形式来代表规则，它等价于前面图3-5中用文字来代表的规则。缺省值鸢尾花 setosa 是进入节点，位于左上方。水平虚线路径指向的是例外，因此接下来的方框，内含结论为鸢尾花 versicolor的一条规则，便是缺省值的一个例外。在它的下方是另一个选择项，第二个例外，选择项路径用垂直实线导致结论是鸢尾花 virginica 。上方的路径沿水平方向通向鸢尾花 versicolor规则的一个例外，只要右上方框中的条件成立，便用结论鸢尾花 virginica 推翻鸢尾花 versicolor规则结论。这个例外的下方是另一个选择项，（碰巧）导致相同的结论。回到下方中间的方框，它也有自己的例外，即右下方结论为鸢尾花 versicolor的方框。每个方框右下角所示的数字是规则的“正确率”，它是用符合规则的实例数量除以满足测试条件但结论不一定相符的实例数量来表示。例如，位于上方中间的方框中所列的条件应用于52个实例，其中有49个是属于鸢尾花 versicolor的。这种表示法的长处在于，你能感觉到位于左侧方框中的规则效果非常好；而位于右侧的方框中的只是覆盖少数几个例外的情况。

为了要产生这些规则，将缺省值先定为鸢尾花 setosa，通常是选择数据集中最频繁出现的类。在这里是任意选择的，因为对这个数据集来说，所有的类都正好出现50次；正如图6-7所示，这个缺省“规则”在150种情形中有50种是正确的。然后寻找出能预测其余类的最好规则。在这个例子中是

```
if petal length ≥ 2.45 and petal length < 5.355
   and petal width < 1.75 then Iris versicolor
```

这条规则覆盖了52个实例，其中49个属于鸢尾花 versicolor。它将数据集分裂成两个子集：52个满足规则条件的实例，剩余的98个不满足条件。

先处理前一部分子集，这部分实例的缺省类是鸢尾花 versicolor：只有3个实例例外，这3

个实例正好都属于鸢尾花 virginica。对于这个子集，预测结果不是鸢尾花 versicolor 的最好的规则是：

```
if petal length ≥ 4.95 and petal width < 1.55 then Iris virginica
```

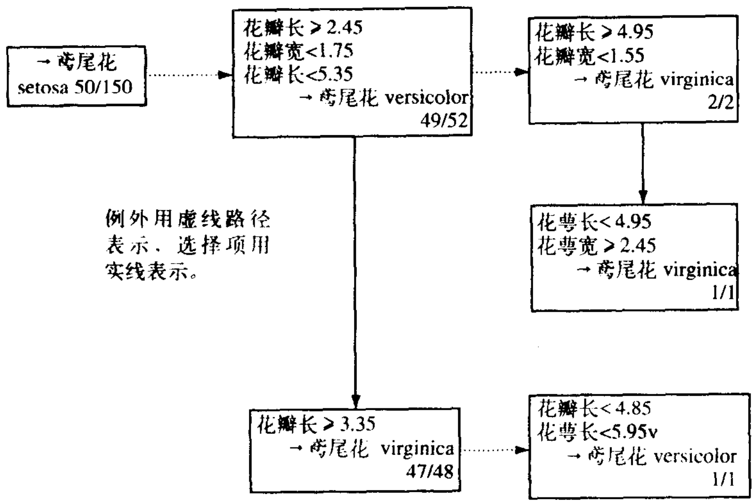


图6-7 鸢尾花数据的包含例外的规则

它覆盖了3个鸢尾花virginica实例中的2个，没有其他的。这条规则将子集再次分裂成两个部分：满足规则条件的和不满足条件的。所幸的是，这次满足条件的实例都是属于鸢尾花 virginica，因此不再需要例外了。但是剩下的实例中还包含着第三个鸢尾花 virginica 实例和49个属于缺省类的鸢尾花 versicolor 实例。再次找到最好的规则：

```
if sepal length < 4.95 and sepal width ≥ 2.45 then Iris virginica
```

这条规则覆盖了剩余的那个鸢尾花 virginica实例，没有其他的，因此也不再需要例外。而且子集中所有其他不满足这个规则条件的实例都是属于鸢尾花 versicolor 类的，这正是当前的缺省类，因此不再需要做什么了。

211～212 现在返回到由初始规则分裂出的第二部分子集，所有不满足下面条件的实例：

```
petal length ≥ 2.45 and petal length < 5.355 and petal width < 1.75
```

对于这些实例，预测类不是缺省值鸢尾花 setosa 的所有规则中最好的是

```
if petal length ≥ 3.35 then Iris virginica
```

它覆盖了这个样本集所含的所有47个鸢尾花 virginica实例（如前面所解释的，另外3个被第一条规则去除了）。它还包含一个鸢尾花 versicolor 实例。这就需要用最后那条规则作为例外处理：

```
if petal length < 4.85 and sepal length < 5.95 then Iris versicolor
```

幸运的是，凡是不满足规则条件的实例都是属于缺省类鸢尾花setosa 。因此结束整个程序。

所产生的这些规则有个特点，即大多数的实例都被高层的规则所覆盖，而低层的规则正是代表例外。例如，先前规则中的最后一个例外条款以及嵌套的else条款的最深层都只覆盖了单个实例，如将它们去除几乎不会有什么影响。即使是其余的嵌套内的例外规则也只是覆盖了2个实例。因此人们可以忽略所有深层结构而只关心前面的一、二层便能对这些规则到底干

什么能有非常清楚的认识。这正是包含例外规则的诱人之处。

6.2.7 讨论

以上所讨论的所有产生分类规则的算法都是使用基本的覆盖或割治方案。对于简单而无干扰的情形，所使用的是一种简单易懂的算法，称为PRISM（Cendrowska，1987年）。当应用于有闭合假设的二类问题，只需要对其中的一个类建立规则，那样的规则属于析取范式，应用于测试实例时不会产生模棱两可的情形。当应用于多类问题，则为每个类生成一个规则集，这样一个测试实例可能被赋予一个以上的类别，或者一个也没有，如果是要寻找单一的预测类的话，有必要考虑进一步的解决方法。

为减少干扰情形下的过度拟合问题，有必要产生一些甚至在训练集上也不"完美"的规则。为了做到这点，需要对规则的"良好度"或价值进行衡量。有了这种衡量，才有可能放弃基本的覆盖算法所使用的一个类接一个类的逐个进行的方法，而采用以产生最好的规则为开端，无论它是对哪个类别的预测，然后将这个规则所覆盖的所有实例去除，再继续这个过程。这形成了产生决策列的方法，而不是产生一系列独立的分类规则。决策列的重大优点是用它们做判定说明时不会产生一些不明确的结论。 213

递增减少-误差修剪法的思想要归功于Fürnkranz和Widmer（1994年），为快速且有效的规则归纳法的形成奠定了基础。RIPPER规则学习器是由Cohen（1995年）提出的，尽管公开发表的论述在关于描述长度DL怎样影响停止条件问题上，看起来与实现时有所不同。这里呈现的只是算法的一些基本理念；在实现时还有太多的细节。

衡量一条规则的价值问题至今还未得到满意的解决。人们提议了许多不同的方法，一些是属于探索式的，另一些基于信息理论或概率。然而，至于哪种才是最好的方案还没有达成统一意见。Fürnkranz和Flach对于各种不同标准做了深入的理论研究（即将完成）。

基于局部决策树的规则学习方法是由Frank和Witten（1998年）创建的。它所产生的规则集的正确率等同于由C4.5所产生的结果，并且比其他一些快速规则归纳法的正确率更高。然而，它的主要优点不在于性能上，而在于它的简单性：将自上而下的决策树归纳法和割治规则学习法结合起来，它不需要进行全局优化却能获得好的规则集。

产生包含例外的规则的程序是由Gaines和Compton（1995年）在创建归纳系统时提出的一种主张，称为 ripple-down 规则。在对一个大型的医学数据集（22 000 个实例，32个属性，60个类别）做试验时，他们发现用含例外的规则来表现大型系统，相比用普通规则来表现等价的系统更易于理解。因为这正符合人们对于复杂的医学诊断所使用的思考方式。Richards 和 Compton（1998年）把它们描述成是典型知识工程的另一种方法。

6.3 扩展线性模型

第4.6节中介绍了如何在所有属性都为数值型的情形下，使用简单的线性模型进行分类。它们最大的缺陷是只能描绘存在于类之间的线性边界，对于很多实践应用来说，这未免太过简单了。支持向量机能利用线性模型来实现对非线性分类边界的描绘。（虽说支持向量机是一个应用广泛的术语，却有点用词不当，实际上是算法，而不是机器。）这怎么可能呢？诀窍很简单，将输入进行非线性映射的转换，换句话说，将实例空间转换为一个新的空间。由于用 214
了非线性映射，在新空间里的一条直线，在原来的空间里看起来却不是直的。在新空间里建

立的线性模型可以代表在原来空间里的非线性决策边界。

想象一下在4.6节中介绍的普通线性模型中直接应用这个想法。例如，原先的属性集可以被替换成它们所能组成的所有n次乘积所形成的属性集。举例来说，对于两个属性，所能组成的所有3次乘积便是

$$x = w_1a_1^3 + w_2a_1^2a_2 + w_3a_1a_2^2 + w_4a_2^3$$

这里x是结果，a_1和a_2是两个属性值，有4个权值w_i需要进行学习。如4.6节所述，结果可用于分类。针对每个类各训练出一个线性系统，将一个未知的实例归到输出结果值x最大的类，这是多反馈线性回归的标准方法。a_1和a_2是测试实例的属性值。为了要在这些乘积所跨越的空间上形成一个线性模型，每个训练实例通过计算它的两个属性值所有可能的3次乘积，被映射到新的空间。学习算法也因此被应用到转换了的实例集上。为了要对一个实例进行分类，在分类前也要对实例先进行相同的转换处理。没有什么可以阻止我们增添更多的合成属性。比如，如果再包括一个常数项、原始属性自身以及所有的2次乘积，总共将产生10个需要学习的权值。（或者，增加一个额外的属性，属性值永远是一个常数，效果也是一样的。）实际上，达到足够高的次数的多项式能以任何要求的精确度接近决策边界。

似乎好得难以置信，的确如此。你也许已猜测到，由于现实设置的转换引入的大量系数会使这个过程出现问题。第一个障碍是计算复杂度。假设原始数据集中有10个属性，我们要包含所有的5次乘积，那么这个学习算法必须要确定2000多个系数。如果对于线性回归，运行时间以属性数量的立方来计算，训练将无法实现。这是一个实用性问题。第二个问题是个基本问题，过度拟合。如果相对于训练实例的数量系数数目过多，那么所形成的模型会“过度非线性化”，将对训练数据过度拟合。模型中的参数实在是太多了。

6.3.1 最大边际超平面

支持向量机能解决这两个问题。支持向量机是基于寻找一种特别的线性模型：最大边际超平面（maximum margin hyperplane）的算法。我们已经知道了超平面，它只是线性模型的
215 另一个用词。为了从视觉上感受一下最大边际超平面，想象有一个含有两个类的数据集，这两个类是可以被线性分隔的；也就是说在实例空间上存在一个超平面能对所有的训练实例进行正确分类。最大边际超平面是一个能使两个类之间达到最大限度分离的超平面，它对两个类都是尽可能地不靠近。图6-8中给出了一个例子，两个类分别用空心圈和实心圈来表示。从技术上说，一系列点的凸包即最紧凑的凸多边形，它的轮廓是通过将每个点与其他所有的点连接而形成的。我们已经假设这两个类是线性分隔的，它们的凸包不能重叠。在所有能分隔这两个类的超平面中，最大边际超平面是其中离两个凸包距离尽可能远的那个，它垂直于两个凸包间距离最短的线段（见图中的虚线）。

最靠近最大边际超平面的实例即距离超平面最近的实例，称为支持向量。每个类至少有一个支持向量，经常是多个。重要的是支持向量能为学习问题定义独特的最大边际超平面。给出两个类的支持向量，我们可以很容易地建造最大边际超平面。其他的训练实例都是无关紧要的，即使被去除也不会改变超平面的方位。

在含有两个属性的情况下，一个分隔两个类的超平面可表达为

$$x = w_0 + w_1a_1 + w_2a_2$$

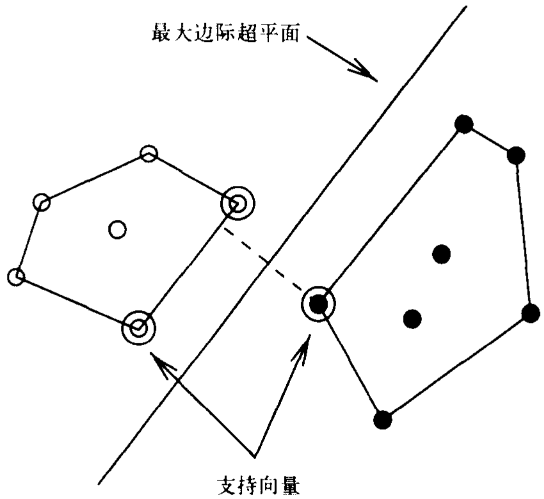


图6-8 最大边际超平面

216

其中a_1和a_2是两个属性值，有3个权值w_i需要学习。然而，定义最大边际超平面的等式也可以从支持向量的角度，用另一种形式表示。训练实例的类值y是1（表示yes，属于这个类），或者－1（表示no，不属于这个类）。那么最大边际超平面就是

$$x = b + \sum_{i\text{是支持向量}} a_i\, y_i\, \mathbf{a}(i) \cdot \mathbf{a}$$

这里，y_i是训练实例$\mathbf{a}(i)$的类值；b和α_i是需要由学习算法来决定的数值参数。注意$\mathbf{a}(i)$和$\mathbf{a}$是向量。向量$\mathbf{a}$代表一个测试实例，正如在前面公式中向量$[a_1, a_2]$代表一个测试实例一样。向量$\mathbf{a}(i)$是支持向量，即图6-8中被圈起来的点，它们是从训练集中挑选出的。$\mathbf{a}(i) \cdot \mathbf{a}$代表了测试实例和一个支持向量的标量积（点积）。如果你对标量积的概念不熟悉，你依然能理解随后的要点，只要把$\mathbf{a}(i)$看作是第i个支持向量的整个属性值系列。最后，b和α_i是决定超平面的参数，就像在前面一个公式中w_0、w_1和w_2是决定超平面的参数。

寻找实例集的支持向量，并决定参数b和α_i，属于标准的优化问题，称为受限的二次优化（constrained quadratic optimization）。有现成的软件可用来解决这些问题（参见Fletcher, 1987年,介绍了全面、实用的解决方法）。然而，如果采用特别的算法来训练支持向量机，可降低计算复杂度、加速学习过程，但这些算法的详细内容不在本书讨论的范围内（Platt，1998年）。

6.3.2 非线性类边界

我们是因为断言支持向量机可用于模拟非线性分类边界而引入介绍支持向量机的。但到目前为止，我们只讲述了线性的情况。考虑一下在决定最大边际超平面之前，对训练集数据进行如前所述的属性转换将会发生什么。直接应用这种转换来建立线性模型，存在两个问题：一个是无法实现的计算复杂度，另一个是过度拟合。

使用支持向量，过度拟合不太可能发生。过度拟合的原因是与不稳定性密切相关的，改动一个或两个实例向量会引起大范围的决策边界的变化。但最大边际超平面则相对比较稳定： 217
只有当被增加或去除的训练实例是支持向量时，边界才会变动。即使在经非线性转换的高维空间也是如此。过度拟合的起因是决策边界过分适合。支持向量是整个训练点集的全局代表，

通常只是它们中的极小部分，几乎不具适合性。因此过度拟合不太可能发生。

计算复杂度又如何呢？这仍然是个问题。假设转换的空间是高维空间，因此转换的支持向量和测试实例由许多分量组成。根据上面的等式，每次对一个实例分类时，必须计算它和所有支持向量的标量积。在经非线性映射所得到的高维空间上，这是相当耗费的。要得到标量积，涉及到对每个属性进行乘积运算和求和运算，并且新空间的属性数目可能是庞大的。问题不仅仅发生在分类过程中，在训练过程中也存在这个问题，因为优化算法必须非常频繁地进行同样的标量积计算。

幸运的是，可以在进行非线性映射之前，对原始属性集的标量积进行计算。上述等式的一个高维表达式便简化为

$$x = b + \sum \alpha_i \, y_i \, (\mathbf{a}(i) \cdot \mathbf{a})^n$$

这里n是所选的转换系数（我们先前的例子中用的是3）。如果将$(\mathbf{a}(i) \cdot \mathbf{a})^n$项展开，将发现它包含了当测试和训练向量按涵盖所有可能的n次乘积转换，然后取标量积作为结果，所涉及的所有高维项。(如果做计算，你会注意到一些常量因子，即二项式系数被引入。但这不要紧，我们关注的是空间维度数，常量只是关系到数轴刻度比例。) 由于这个在数学上的等效关系，标量积可以在原先的低维数空间上进行计算，因此，问题便解决了。在实现时，可选用受限的二次优化软件包，将每次求$\mathbf{a}(i) \cdot \mathbf{a}$的值替换为求$(\mathbf{a}(i) \cdot \mathbf{a})^n$的值。就是这么简单，因为在优化和分类算法中，这些向量都只是以标量积形式参与。训练集向量，包括支持向量，以及测试实例在计算过程中都保留在原来的低维空间里。

函数$(\mathbf{x} \cdot \mathbf{y})^n$，计算向量$x$和向量$y$的标量积并将结果上升为$n$次幂，称为多项式核
218 (polynomial kernel) 函数。选择n的好方法是从1（一个线性模型）开始，然后递增，直到估计误差不再有改进为止。通常，相当小的数目就足够了。

也可以换用其他的核函数来实现不同的非线性映射。两个经常建议使用的是径向基核函数（radial basis function (RBF) kernel）和S形核函数（sigmoid kernel）。哪一种能产生最好的效果取决于应用，但在实践中它们的差别并不很大。有趣的是使用径向基核函数的支持向量机只是神经网络（neural network）的一种，称为径向基函数网络（我们将在下面讨论），而使用S形核函数的支持向量机实现的是另一种神经网络，即带一个隐蔽层的多层感知器（也将在下面讨论）。

在整节中，我们都假设训练数据是可以线性分隔的，在实例空间或在经过非线性映射的新空间。结论是支持向量机可以推广到训练数据不可分隔的情形。最终的实现要给上述系数α_i设定一个上限。不幸的是，这个参数需要用户自己选择，而最好的设置只有从实验中得出。并且在极大多数情况下，不可能事先判定一个数据集是否可被线性分隔。

最后我们必须提醒的是，和其他的学习方法（如决策树）相比较，即使是最快的支持向量机训练算法应用在非线性情形下也是较慢的。但另一方面，由于它们能得到微妙复杂的决策边界，因此通常能产生精确度很高的分类器。

6.3.3 支持向量回归

最大边际超平面的概念只能应用于分类。但是，支持向量机算法已发展成能适用于数值预测，它共享了许多在分类应用中所遇到的特性：产生一个通常只是由少许支持向量来表示

的模型，并可以利用核函数将其应用于非线性问题中。如介绍普通的支持向量机一样，我们只讨论它所引入的概念，而不具体描述算法实际是如何工作的。

和4.6节中讨论的线性回归一样，基本理念是通过使预测误差最小化来寻找一个函数能较好地接近训练数据点。主要的差别在于所有在用户指定参数ε之内的偏差都被简单地丢弃了。 219
在进行误差最小化时，由于同时试图使函数平面度（flatness）最大化，从而过度拟合的风险降低了。另一个差别在于被最小化的通常是预测值的绝对误差，用以替换线性回归中所用的误差平方。（但是也有使用误差平方的算法版本。）

用户指定参数ε是在回归函数周围定义一个管道，在这个管道内的误差将被忽略：对线性支持向量回归来说，这个管道是个圆柱体。如果所有的训练点都在宽度为2ε的管道内，算法输出一个位于最平的管道中央的函数，这个管道包含所有训练点。这个情况下察觉到的总误差为0。图6-9a展示了一个含有1个属性、1个数值型类和8个实例的回归问题。在这个例子中，ε设为1，因此回归函数周围管道宽度（图中由虚线画出）为2。图6-9b展示的是当ε设为2时的学习结果。正如你所见，较宽的管道可能产生一个较平的函数。

ε的值控制了函数与训练集数据的拟合程度。这个值太大将产生一个毫无意义的预测器。极端的情形是，当2ε超出训练数据的类的数值范围时，回归线是水平的，并且算法的预测结果总是类的平均值。另一方面，ε值太小，也许不存在能包含所有训练数据的管道。在这种情况下，部分训练数据点将出现非零误差，这里预测误差和管道的平面度之间就存在一个折中问题。在图6-9c中，ε设为0.5，而没有一个宽度为1的管道可以包含所有的数据。

在线性情况下，支持向量回归函数可以写成

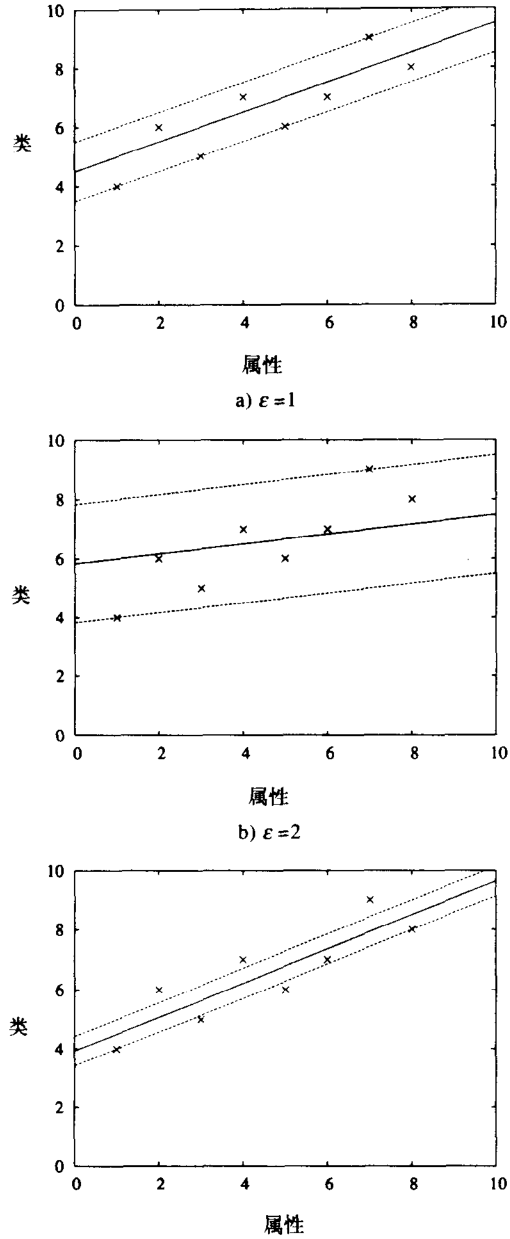
$$x = b + \sum_{i\text{ 是支持向量}} \alpha_i \, \mathbf{a}(i) \cdot \mathbf{a}$$

如同分类问题一样，在处理非线性问题时，标量积可以用核函数来代替。支持向量是所有那些不是确实落入管道内的点，即那些在管道外及管道边缘上的点。如分类问题一样，所有其他点的系数为0，都可以从训练集中去除而不改变学习结果。和分类问题对比，这里α_i可能是负的。

正如我们先前提到的，在使误差最小化的同时，算法还要试图使回归函数平面度最大化。在图6-9a和图6-9b中，存在包含所有训练数据点的管道，而算法只是输出其中最平的管道。但在图6-9c中，不存在误差为0的管道，在预测误差和管道平面度之间要进行权衡。这个权衡是通过对系数α_i的绝对值设定上限值C来控制。这个上限值约束了支持向量对回归函数形状的影响，它是除了ε外，必须由用户设定的另一个参数。C值越大，函数越是与数据拟合。在$\varepsilon=0$ 220
的情况下，算法只是简单地实行在系数受限条件下的最小绝对误差回归，所有的训练数据都成为支持向量。相反，如果ε足够大而能使管道包含所有的训练数据，误差为0，不需要进行权衡，算法输出的是包含数据且与C值无关的最平的管道。

6.3.4 核感知器

在4.6节中，我们介绍了用于学习线性分类器的感知器（perceptron）算法。核函数的奥妙也可以应用于这个算法，使之升级用以学习非线性决策边界。为了能看到这一点，我们先来复习一下线性的情况。感知器算法让训练集实例一个接一个进行重复迭代，每当其中的一个实例按目前所学习的权向量进行分类出现错误时，便要更新权向量。更新只是简单地在权向



a) $\varepsilon=1$

b) $\varepsilon=2$

c) $\varepsilon=0.5$

图6-9　支持向量回归

量上加上或减去这个实例的属性值。这意味着最终的权向量正是被错误分类的实例的总和。感知器做预测是基于

$$\sum_i w_i a_i$$

大于还是小于0，其中w_i是第i个属性的权值，a_i是我们要进行分类的实例所对应的属性值。

我们也可以用

$$\sum_i \sum_j y(j)a'(j)_i a_i$$

这里，$a'(j)$是第j个被错误分类的训练实例，$a'(j)_i$是它的第i个属性值，$y(j)$是它的类值（不是+1就是-1）。在实现时，我们不再需要清楚地记录权向量，只需要保存当前被错误分类的实例，然后利用上面的表达式来预测。

看来似乎没有什么可取之处，但事实上，这个算法相当慢，因为每做一次预测需要重复处理所有的错误分类实例。然而，再仔细地看一下这个公式，发现它可以用标量积来表示。首先交换一下总和符号得到

$$\sum_j y(j) \sum_i a'(j)_i a_i$$

第二个总和正是两个实例之间的标量积，可以写成

$$\sum_j y(j)\mathbf{a}'(j)\cdot \mathbf{a}$$

关键在此！类似于支持向量机的表达式使我们能利用核函数。确实，在这里我们可以采取完全相同的妙计，利用核函数来替换标量积。将这个函数表示为$K(\cdots)$，得到 221~222

$$\sum_j y(j)K(\mathbf{a}'(j),\mathbf{a})$$

这样，感知器算法只要简单地记录下在训练过程中被错误分类的训练实例，用上面的表达式来做预测，便能学习非线性分类器。

如果利用核函数，在高维空间存在一个分隔超平面，利用这个算法会得到一个。但是，所得到的不是支持向量机分类器所能得到的最大边际超平面。这意味着它的分类性能通常比较差。从好的方面看，这个算法容易实现并且支持递增学习（incremental learning）。

这个分类器称为核感知器（kernel perceptron）。结论是所有学习线性模型的算法都可以通过应用核函数的奥妙之处进行升级。例如，logistic 回归可以转变为核logistic 回归。同样适用于回归问题：线性回归也可以利用核函数来升级。这些高级的线性回归和logistic回归方法的缺点（如果它们是采取直接应用）在于解决方案不是“稀疏的”：每个训练实例对最终作为解决方案的向量都有贡献。在支持向量机和核感知器中，只是部分训练实例影响最终的问题解决方案，这点在计算效率上会造成很大的差别。

感知器算法所得到的作为解决方案的向量在很大程度上依赖于实例的顺序。一种可以使算法比较稳定的方法是使用在训练过程中所经历到的所有权向量，不单单是最终的那个，让它们对预测进行投票。每个权向量贡献一定数目的票数。直观地来看，一个权向量的“正确性”可以粗略地用它自形成后对随后的训练实例进行正确分类而无需更新的逐个试探数目来衡量，可以将衡量所得的数目赋予这个权向量作为它要贡献的票数，这个算法便是投票感知器（voted perceptron），它的性能与支持向量机相差无几。（注意，如前所述，投票感知器中的不同权向量也不需要记录，核函数在这里也同样适用。）

6.3.5 多层感知器

为建立一个基于感知器的非线性分类器，使用核函数并不是唯一的方法。实际上，核函数是近期才开发出来的机器学习方法。之前，神经网络对于非线性分类问题采用另一种不同 223

的方法：它们将许多简单的类似感知器的模型按层次结构连接起来。这样就能表现非线性决策边界。

在4.6节中曾解释过感知器代表一个存在于实例空间的超平面。我们曾提到它有时被描述为人造“神经元”。当然，人和动物的大脑能成功地完成非常复杂的分类问题，例如，图像识别。单靠大脑中单个神经元的功能是不足以完成这些壮举的。类似大脑的结构是如何解决问题的呢？答案在于大脑神经元是大规模地相互连接的，它能将一个问题分解为可以在神经元这一层来解决的子问题。这个观察结果启发了人工神经网络的发展。

考虑图6-10中一个简单的数据集。图6-10a展示了一个二维实例空间，含有4个实例，实例的类为0和1，分别用白的和黑的圆点来表示。无论你怎样在这个空间上画直线，都无法找到一条直线能将黑白圆点分隔开来。换句话说，这个问题不是能用线性分隔的，简单的感知器算法无法生成分隔超平面（在二维实例空间，超平面只能是一条直线）。图6-10b、6-10c的情形就不同了，这两种情况都是可以线性分隔的。图6-10d的情况也是如此，图中展示了在一维实例空间中的两个实例点（在一维实例空间，超平面退缩成一个分隔点）。

如果对命题逻辑熟悉的话，你也许会注意到图6-10中的四种情况正好对应四种逻辑关系。图6-10a 代表逻辑异或（XOR），当且仅当有一个属性值为1时，类值为1。图6-10b代表逻辑与（AND），当且仅当两个属性值都为1时，类值为1。图6-10c代表逻辑或（OR），当且仅当两个属性值都为0时，类值为0。图6-10d代表逻辑非（NOT），当且仅当属性值为1时，类值为0。由于后面三种情况都是可以线性分隔的，因此感知器可以代表逻辑与、逻辑或及逻辑非。图6-10f~h分别展示了与数据集相对应的感知器。可是简单的感知器不能代表逻辑异或（XOR），因它不能被线性分隔。解决这个问题的分类器仅用一个感知器是不够的，需要几个感知器。

图6-10e展示了一个由三个感知器，或称单元（unit），标记为A、B和C 所组成的网络。前
224 面两个与网络输入层（input layer）相连接，输入层代表数据的属性。就像在简单的感知器中一样，输入层还附加一个称为偏差（bias）的常量输入。然而，第三个单元却和输入层没有任何连接。它的输入是由单元A、B的输出（非0即1）和另一个常量偏差单元所组成。这三个单元构成了多层感知器的隐蔽层（hidden layer）。称为“隐蔽”是因为单元与环境没有直接的连接。正是这一层让系统代表异或(XOR)功能。可以用所有4种可能的输入信号组合来进行核实。例如，如果属性a_1值为1，a_2值也为1，那么单元A将输出1（因为$1\times1+1\times1-0.5\times1>0$），单元B 将输出0（因为$-1\times1+(-1)\times1+1.5\times1<0$），单元C将输出0（因为$1\times1+1\times0+(-1.5)\times1<0$）。这是正确的答案。再仔细检查一下这三个单元的功能发现，第一个代表或，第二个代表非与（NAND）（逻辑非NOT和逻辑与AND组合在一起），第三个是与。组合在一起代表了（a_1 OR a_2）AND （a_1 NAND a_3），这正是异或的精确定义。

正如这个例子所展示的，命题计算的任何表达式都可以转化为一个多层感知器，因为逻辑与、逻辑或和逻辑非这三种连接关系就已足够用了，并且我们也已经看到了感知器是如何来代表每种关系的。个体单元可以连接在一起来表示任意复杂的表达式。因此多层感知器也具备相同的表达能力，譬如说，与决策树相比。实际上，一个两层的感知器（不算输入层）就足够了。这时，隐蔽层的每个单元对应不同的逻辑与，不同是因为假设它可能在做逻辑与之前，部分输入要进行逻辑非操作；然后在输出层进行逻辑或操作，它是由单个单元来表示的。换句话说，隐蔽层的每个单元所担当的角色就如同叶节点在决策树中，或是单个规则在决策规则集中所担当的角色。

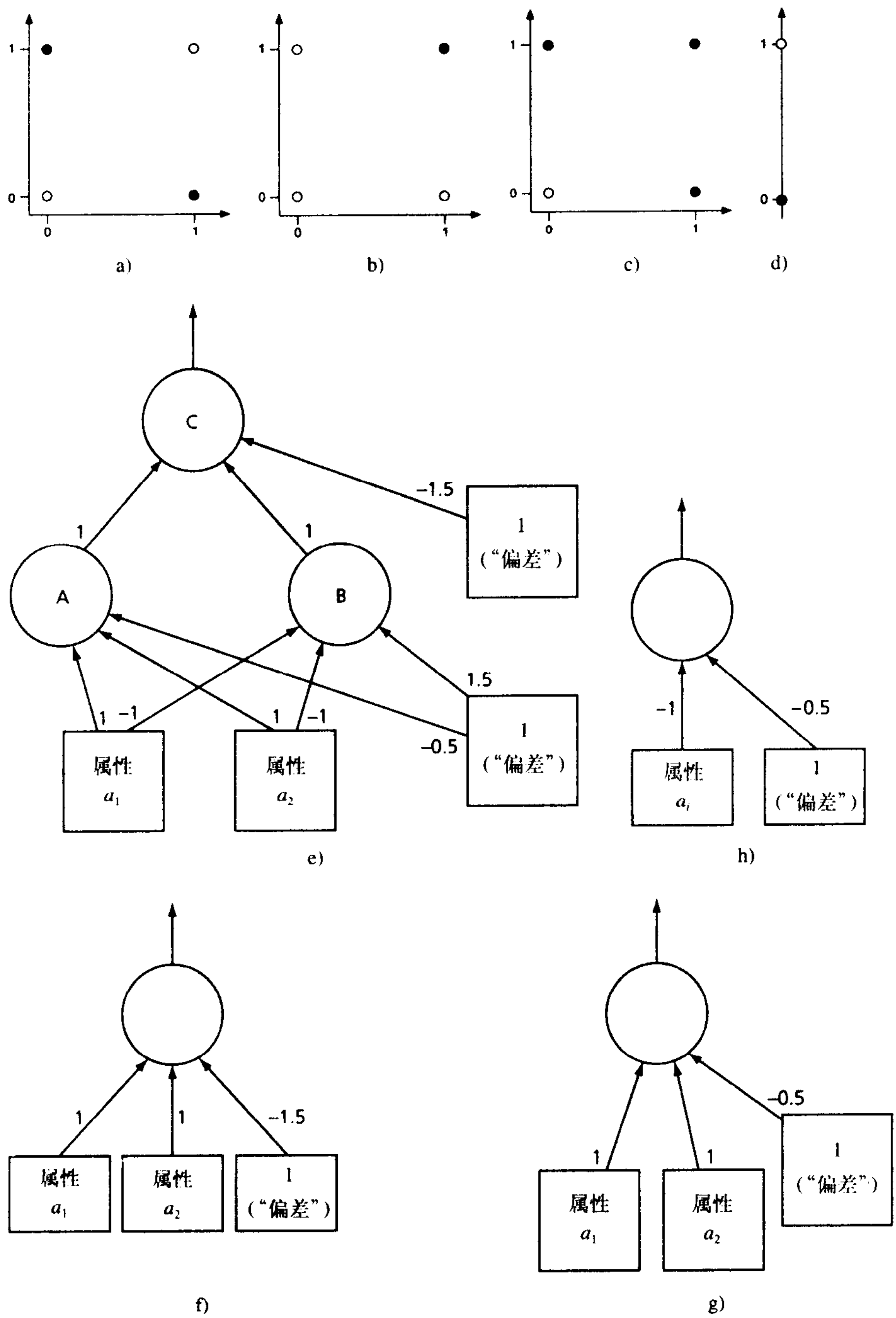


图6-10 数据集样例及其对应的感知器

重要的问题是怎样来学习多层感知器。这个问题有两个方面：一方面是要学习网络结构，另一方面是要学习连接权值。对于一个给定的网络结构，有一种相当简单的算法可用来决定权值。这个算法称为反向传播（backpropagation）法，将在下节中讨论。虽然现有许多试图识别网络结构的算法，但这方面的问题通常是通过实践来解决的，也许需要结合

225~226 一些专家知识。有时可以将网络分隔成不同的模块，代表不同的子任务（例如，在图像识别问题中，辨别一个物体不同的组成部分），这开创了一种将领域知识与学习过程结合起来的方法。通常一层隐蔽层便是所需的全部，而隐蔽层适合的单元数目则是通过估计正确率最大化来决定的。

6.3.6 反向传播法

假设现有一些数据，我们要寻找一个多层感知器，它是潜在分类问题的正确预测器。已知固定的网络结构，必须要决定合适的网络连接权值。在没有隐蔽层的情况下，可以用4.6节中的感知器学习规则来找到合适的值。但是现在假设有隐蔽层。要是知道输出单元该预测什么，就能根据感知器规则来调整连接这个单元的权值。可是连接到隐蔽层单元的正确输出结果是未知的，因此感知器规则在这里不适用。

一般来说，解决办法是根据每个单元对最终预测的贡献调整连接隐蔽单元的权值。有一种称为梯度下降法（gradient descent）的标准数学优化算法能达到此目的。不幸的是，需要求导数，而简单的感知器使用阶梯函数来将加了权的输入总和转换为0/1预测，阶梯函数是不可微的。因此必须考虑是否能将阶梯函数替换为其他形式的函数。

图6-11a展示了阶梯函数：如果输入小于0，输出为0，否则输出为1。我们需要找一个形状类似，但可微的函数。通常使用的替代函数是图6-11b所展示的函数。在神经网络术语中，称为S形函数并定义为

$$f(x)=\frac{1}{1+e^{-x}}$$

在4.6节讨论logistic 回归中使用对数变换（logit transform）时，曾遇到过S形函数。实际上，学习多层感知器与logistic 回归是密切相关的。

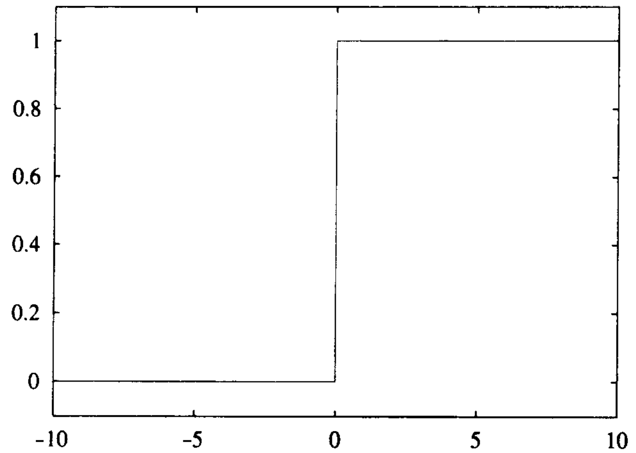
为了要应用梯度下降程序，通过调整权值使误差函数达到最小化，也必须是可微分的。用5.6节中介绍的离散的0-1损失函数来衡量错误分类数目，也不符合这个标准。作为替代，在多层感知器训练时，一般是将网络输出的误差平方最小化，并在本质上把它看作是对类概率的估计。（其他损失函数也可以用。例如，如果用似然函数代替误差平方，学习S形的感知器和logistic回归是一样的。）

227 我们采用误差平方损失函数，因它的应用最为广泛。对单个训练实例来说，它是

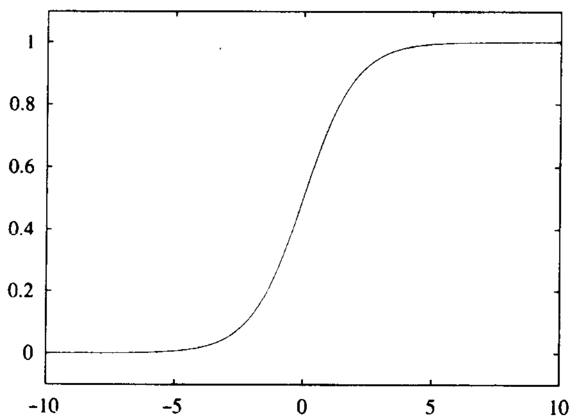
$$E=\frac{1}{2}(y-f(x))^2$$

这里，$f(x)$ 是从输出单元得到的网络预测，y是实例的类标记（这里假设不是1就是0）。系数1/2只是为了方便起见，当开始计算导数时就去掉。

梯度下降揭示了将被最小化的函数，这里是误差函数的导数所提供的信息。举个例子，假设误差函数正好是x^2+1，如图6-12所示。X轴代表一个要进行优化的假设参数。x^2+1的导数是$2x$。重要的是，根据导数可以计算出函数在任意点的斜率。如果导数是负的，函数向右下方倾斜；如果是正的，则向左下方倾斜；导数的大小决定了倾斜的陡峭度。梯度下降法利用这些信息来调整函数的参数，它是一个迭代优化的过程。它将导数值乘以一个称为学习率（learning rate）的小常量，然后将计算结果从目前的参数值中减去。代入新的参数重复这个过程，以此类推，直到达到最小值。



a) 阶梯函数



b) S 形函数

图6-11 阶梯对照S形

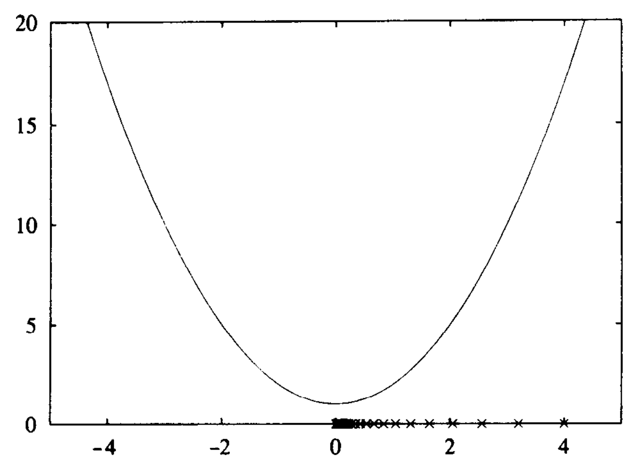


图6-12 误差函数x^2+1的梯度下降

回到上述例子中，假设学习率设定为0.1，目前的参数值x为4。导数便是它的两倍，就是8。乘以学习率得到0.8，再将其从4中减去，得到3.2，这便是新的参数值。用3.2作为参数值再重
228~229 复这个过程，得到2.56，然后又得到2.048，等等。图6-12中的小叉显示的就是在这个过程中所产生的数值。一旦参数值变化变得非常小，就停止这个过程。在此例子中，这个情况发生在当参数值逼近于0时，这个值对应于X轴上的位置正是假设误差函数达到最小值的位置。

学习率决定了改变的大小，因此决定了多快能使搜索达到收敛。如果这个值太大，误差函数又有几个最小值，搜索也许会越过目标，错过最小值，或者出现大幅振荡。如果太小，靠近最小值的进程可能会很慢。要注意的是，梯度下降法只能找到局部最小值。如果函数有几个最小值，多层感知器的误差函数通常有多个最小值，找到的也许不是最好的。这是标准的多层感知器与其他方法如支持向量机相比的一个最明显的缺陷。

为了利用梯度下降法来寻找多层感知器的权值，误差平方的导数必须根据每个参数来决定，即在网络中的每个权值来决定。先来看一下不带隐蔽层的简单感知器。根据某个具体的权值w_i，对上述误差函数求导，得到

$$\frac{dE}{dw_i}=(y-f(x))\frac{df(x)}{dw_i}$$

这里$f(x)$是感知器的输出，x是加了权的输入总和。

要计算等式右边第二个因子，需要对S形函数求导。得到的是可以用函数$f(x)$自身来表示的特别简单的形式：

$$\frac{df(x)}{dx}=f(x)(1-f(x))$$

用$f'(x)$表示这个导数。可目的是要对w_i求导而不是对x求导。因为

$$x=\sum_i w_i a_i$$

所以$f(x)$对w_i求导是

$$\frac{df(x)}{dw_i}=f'(x)a_i$$

将这个结果代入对误差函数求导，得到

$$\frac{dE}{dw_i}=(y-f(x))f'(x)a_i$$

这个表达式给出了改变权值w_i计算所需的全部，这个改变是由于某个具体的实例向量**a**（如前所述，再附加一个偏差1）所引起的。对每个训练实例重复进行这样的计算，然后把改变值按
230 具体的w_i分别相加，乘以学习率，再从目前的w_i值中减去计算结果。

到现在为止都还不错。但是所有这些都是在假设没有隐蔽层的前提下。带有隐蔽层就稍许难对付一些。假设$f(x_i)$是第i个隐蔽单元的输出，w_{ij}是从输入j到第i个隐蔽单元中间连接的权值，w_i是第i个隐蔽单元和输出单元之间的权值。图6-13描绘了这个情形。和前面一样，$f(x)$是输出层的单个输出单元。权值w_i的更新规则和上述方法基本相同，除了用第i个隐蔽单元的输出替换了a_i：

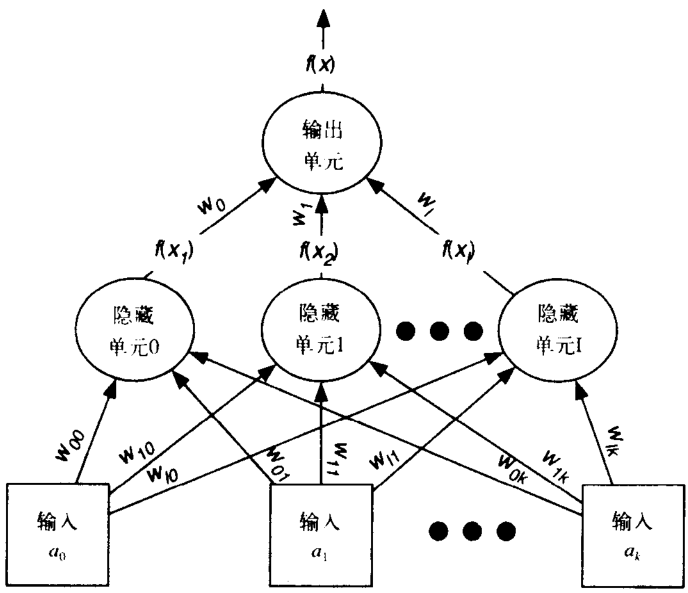


图6-13 带一层隐蔽层的多层感知器

$$\frac{dE}{dw_i}=(y-f(x))f'(x)f(x_i)$$

然而，要更新权值w_{ij}，需要进行相应的求导计算。应用链规则得出

$$\frac{dE}{dw_{ij}}=\frac{dE}{dx}\frac{dx}{dw_{ij}}=(y-f(x))f'(x)\frac{dx}{dw_{ij}}$$

前两个因子与先前的公式是一样的。要计算第三个因子，需要进一步求导。因为 231

$$x=\sum_i w_i f(x_i)$$

$$\frac{dx}{dw_{ij}}=w_i\frac{df(x_i)}{dw_{ij}}$$

而且，

$$\frac{df(x_i)}{dw_{ij}}=f'(x_i)\frac{dx_i}{dw_{ij}}=f'(x_i)a_i$$

这意味着问题解决了。将所有这些归纳在一起，得到误差函数对权值w_{ij}求导的公式：

$$\frac{dE}{dw_{ij}}=(y-f(x))f'(x)w_i f'(x_i)a_i$$

同前面一样，对每个训练实例进行这样的计算，然后把改变值按具体的w_{ij}分别相加，乘以学习率，再从目前w_{ij}值中减去计算结果。

这个求导方法是应用于含一层隐蔽层的感知器。如果存在两层隐蔽层，可以再次应用相同的策略来更新第一隐蔽层的输入连接权值，错误从输出单元通过第二隐蔽层传播到第一层。由于这样一种错误传播机构，这种普通的梯度下降策略称为反向传播法。

默认假设是网络输出层只有一个单元，适合于二类问题。对含有两个以上类的情况，可

以将每个类和其他的类区分开来进行一个个单独的网络学习。也可以通过为每个类各建立一个输出单元，将隐蔽层的每个单元和每个输出单元相连接，从而能在单个网络上获得一个更为紧凑的分类器。某个具体训练实例的误差平方是所有输出单元的误差平方的总和。相同的技术亦可应用于同时预测几个目标或是几个属性值，它是通过为每个（目标或属性值）各建立一个输出单元。从直觉上来看，如果潜在的各项学习任务之间存在联系的话，这种方法可能要比给每个类各建一个分类器的预测正确率高。

我们已经假设权值只是在所有训练实例都传送到网络后才进行更新，所有相应的权值改变是累积起来的。这是批处理学习（batch learning），因为所有的训练数据是在一起处理的。但是也可以用完全同样的公式，在对每一个训练实例进行处理后即更新权值。这叫随机反向
232 传播（stochastic backpropagation）。由于每次更新后，总体误差率不一定下降，因此不能保证它能收敛于最小值处。它可用于在线学习，这种情形下新的数据以连续的数据流形式传送进来，对每个训练实例只处理一次。在这两种反向传播方法中，先将属性标准化使之平均值为0、标准差为1，通常是很有帮助的。在开始学习之前，每个权值被初始化为基于平均值为0的正态分布上的、一个很小的随机选择值。

同其他的学习方案一样，用反向传播法训练出的多层感知器可能会过度拟合，特别是当用于代表潜在学习问题结构的网络远大于实际所需时。人们提出许多修改方案用以避免这个问题。一个非常简单的方法称为提前停止(early stopping)，操作起来就像在规则学习器中的基于减少-误差修剪法：用一个旁置的数据集来决定何时停止反向传播算法的迭代过程。测量在旁置数据集上的错误，一旦错误开始增加即停止，因为这表明与训练实例过度拟合了。另一种方法称为权衰减（weight decay），即在误差函数上加一项惩罚项，它由网络中所有权值总和的平方组成。它试图通过惩罚那些权值较大却不能相应地为降低误差作贡献的权值，来限制网络预测中无关连接的影响。

虽然标准梯度下降法是用于学习多层感知器权值的最简单技术，但并不意味着是最有效的方法。在实践中，这个方法相当慢。经常能带来性能改善的一个诀窍是在更新权值时包含一个动量（momentum）项：在新更改的权值上再加上前次迭代更新量的一小部分数值。这使方向改变上不那么突然，平滑了搜寻过程。更复杂的方法是利用误差函数的二次求导信息，这样能更快地达到收敛。然而，和其他的分类器学习方案相比较，即使采用这样的算法还是很慢。

带有隐蔽层的多层感知器的一个严重缺陷是在本质上难于理解。有几种技术试图从训练好的神经网络中提炼规则。但是，与从数据中直接导出规则集的标准规则学习器相比是否有优势并不很清楚，特别是首先考虑到标准规则学习法一般比学习一个多层感知器要快得多。

虽然多层感知器是神经网络中最为著名的，但还有很多其他提议。多层感知器是属于一种称为前馈网络（feedforward networks）的网络类型，因为它们不包含任何的环，网络的输出仅仅依赖于当前的输入实例。循环（recurrent）神经网络就包含环。由先前输入得到的计算
233 结果被反馈到网络中，使网络具有类似记忆的能力。

6.3.7 径向基函数网络

另一种常用的前馈网络是径向基函数（RBF）网络。不算输入层，它共有两层，与多层感知器的不同之处在于隐蔽单元执行计算。每个隐蔽单元在本质上代表输入空间某个特定的

点，而对于一个给定实例，隐蔽单元的输出或称激活（activation）是由它的点和这个实例，即另外一个点之间的距离决定的。从直觉上来看，这两点越是接近，激活力越强。这是通过非线性转换函数将距离转换为相似度来实现的。钟形的高斯激活函数（activation function）就是为达这个目标的一种常用函数，每个隐蔽单元的高斯激活函数的宽度可能是不同的。隐蔽单元称为径向基函数，因为在实例空间的各点通过一个给定的隐蔽单元对其产生相同的激活，形成一个超球面或椭圆体（在多层感知器中，就是超平面）。

RBF网络的输出层和多层感知器的输出层是相同的：它是隐蔽层输出的线性组合，在分类问题中，并且使用S形函数进行传输。

这样一种网络要学习的参数是：（a）径向基函数的中心和宽度，（b）用于形成隐蔽层输出的线性组合的权。一个优于多层感知器的显著优点是，第一组参数的决定可以独立于第二组参数而仍然能产生正确的分类器。

决定第一组参数的一种方法是使用聚类，而不去看训练实例的类标签。可以应用4.8节中介绍简单的k均值聚类算法，独立地对每个类进行聚类得到k个基函数。从直觉上说，所得的径向基函数代表了实例原型。然后可以来学习第二组参数，第一组参数保持固定不变。这牵涉到学习一个线性模型，用我们讨论过的技术（例如，线性或logistic回归）。倘若隐蔽单元数目比训练实例数目小得多，学习很快就完成了。

RBF 网络的一个缺点是由于在距离计算中所有属性采用相同处理，从而赋予每个属性相同的权值。因此和多层感知器相比，它不能有效地处理无关的属性。支持向量机也存在同样的问题。实际上，用高斯核函数的支持向量机（即“RBF”核函数）是RBF网络的一种特例，即基函数是以每个训练实例为中心的，输出是通过计算最大边际超平面来得到线性组合的。它的效果是只有部分径向基函数有非零的权值，那些代表支持向量的径向基函数。 234

6.3.8 讨论

支持向量机源自统计学习理论的研究（Vapnik，1999年），对它的探索始于Burges（1998年）提供的教学指导。Cortes和Vapnik（1995年）发表了一般概括性的描述，包括将其推广到数据处于不可线性分隔的情形。本书介绍了标准的支持向量回归：Schölkopf 等（1999年）提出了一个不同的版本，用一个参数来代替两个参数。Smola和Schölkopf（2004年）提供了支持向量回归法的深入指导。

Freund和Schapire（1999年）提出了（投票）核感知器。Cristianini和Shawe-Taylor（2000年）对支持向量机和其他一些基于核函数的方法，包括潜在于支持向量学习算法中的优化理论都做了介绍。我们只是掠过了这些机器学习方法的表层，主要是因为下层隐含了高深的数学。利用核函数来解决非线性问题的思路被应用在许多算法中，例如主分量分析（principal components analysis）（将在7.3节中讨论）。核函数从本质上来看是一个带有某种数学特性的相似函数，任何一种结构都可以用核函数来定义，如集合、字符串、树以及概率分布。Shawe-Taylor和Cristianini （2004年）对基于核函数的学习做了详尽的介绍。

有关神经网络的文献很多，Bishop（1995年）为多层感知器和RBF 网络都提供了非常好的介绍。自从支持向量机的出现，人们对神经网络的兴趣似乎减少了，许是因为支持向量机需要调整的参数较少，而能达到相同（甚至更高）的正确率。但是多层感知器有它的优势，它们能学习忽略无关属性，利用k均值训练出的RBF网络可以被看作是寻找非线性分类器的一种快速草率的方法。

6.4 基于实例的学习

在4.7节中已经看到最近邻规则是怎样被应用于基于实例学习的基本形式中的。这个简单的方案存在一些实践问题。第一、对于较大规模的训练数据集，它的速度往往很慢，因为对每个测试实例来说，整个训练集都要被搜寻一遍——除非使用像kD树或球树（ball tree）那样更为复杂的数据结构。第二、碰到干扰数据，性能表现较差。因为测试实例的类是由单个最近邻
235 的实例所决定的，而没有使用任何“取平均值”的技术来帮助消除干扰。第三，当不同的属性对分类结果存在不同程度的影响时，即极端的情形是当某些属性对分类来说完全无关时，性能表现较差。因为在距离公式中所有属性的贡献是均等的。第四，它不能实现明确的推广，尽管在3.8节中提到（并在图3-8中描述）某些基于实例的学习系统确实能实现明确的推广。

6.4.1 减少样本集数量

普通的最近邻规则储存了许多重复样本：完全没有必要保存所有目前所见过的实例。一个简便的变体就是用所见的实例对每个实例进行分类，而只保存被错误分类的实例。这里用了一个术语样本集特指先前已见过的、用以分类的实例集。丢弃被正确分类的实例减少样本集的数量，并被证实是修剪样本数据库的一个有效方法。理想状况是只为实例空间上的每个重要区域储存单个的样本。但是在学习过程早期被丢弃的实例，也许在后期看来是重要的实例，这可能导致预测正确率的降低。随着储存实例数目的增加，模型的正确率也随之得以改善，因此系统错误随之减少。

不幸的是，只储存被错误分类的实例，这个策略在碰到干扰数据时不能正常工作。干扰实例很可能要被错误分类，因此导致所储存的样本集是在累积那些最无用的实例。这个结论很容易通过实验观察到。因此，这个策略只是在探寻有效的基于实例学习方法道路上的一块踏脚石。

6.4.2 修剪干扰样本集

对任何不去抑制干扰样本集的最近邻方案来说，干扰样本集不可避免地会降低性能等级，因为它们会重复地造成对新的实例的错误分类。有两种方法可以解决这个问题。一种是预先决定一个常数k，然后查找k个最近邻的实例来代替查找单个最近邻实例，并将多数类赋予未知实例。问题只是怎样给定合适的常数k。简单最近邻学习对应的k值等于1。干扰越多，k值就应定得越大。一种方法是设定几种不同的k值进行交叉验证，然后选择其中最好的。虽然这样计算时间耗费很多，但通常能达到非常好的预测性能。

第二种方法是监测每个储存样本的性能表现，丢弃表现不好的。这可以通过记录每个样
236 本所做出的正确和错误的分类决策数目来完成。在成功率上要预设两个阈值。当某个样本的性能低于阈值下限，它将从样本集中删除。如果性能超过阈值上限，它将用于测试新的实例。如果性能介于两者之间，它将不参与预测，但只要它是新实例最靠近的样本（假如它的性能记录足够好的话，它就会参与预测），它的成功统计就要被更新，就好像它参与了预测那个新的实例。

为了完成这个任务，要利用5.2节中伯努利程式（Bernoulli process）成功概率的置信边界。回想一下，我们将N次测试中含有S次成功，作为潜在真实成功率p的置信边界的依据。给定某一置信度，比如说5%，可以计算出上、下边界，并有95%的把握p将落入上、下边界之内。

为了把这点运用于决定何时接受某个样本，先假设某个样本已参与n次对其他实例的分类，其中s次是正确的。这使我们能估计出在某一置信度下，这个样本真实成功率的边界。现在假设这个样本的类在N个训练实例中出现c次。这使我们能估计出缺省成功率的边界，这里缺省成功率是指对这个类的实例分类的成功概率，而不考虑其他实例的任何信息。我们主张真实成功率的置信边界下限要高于缺省成功率置信边界的上限。用同样的方法来设计丢弃性能表现差的样本的准则，丢弃的条件是真实成功率的置信边界上限低于缺省成功率置信边界的下限。

选择适当的阈值，这个方案工作得很好。在IB3即“基于实例的学习器版本3”的实现中，决定接受的置信度设为5%，决定丢弃的置信度设为12.5%。低百分率将产生较宽的置信区间，是为更严格的准则设计的，因为这使一个区间的下限要位于另一个区间的上限之上更加困难。决定接受的准则比决定丢弃的准则更为严格，使得一个实例被接受更为困难。丢弃准则不那么严格的原因在于丢弃分类正确率中等偏差的实例几乎不会有什么损失：很有可能以后会有类似的实例来取代它。运用这些阈值能使基于实例的学习方案的性能得到提高，同时大幅减少了储存的样本数目，特别是干扰样本。

6.4.3 属性加权

经过修改的欧几里得距离函数使所有的属性值按比例转换为0到1之间的数值，这在各个属性与结果的相关性等同的领域中能较好地工作。然而，这类领域只是一种例外而非常规。在多数领域中，部分属性是与结果无关的，并且在相关属性中部分属性不如其他属性重要。基于实例学习的下一步改进便是通过动态更新属性权值来递增学习每个属性的相关性。 237

在一些方案中，权值是有类特性（class specific）的，即一个属性可能对某种类比对其他类更为重要。为了配合这一点，对每个类进行描述，使每个类的成员与属于其他类的成员区分开来。这就带来了一个问题，即一个未知的测试实例可能会被赋予几个不同的类，或者一个类也没有，一个我们在讨论规则归纳中很熟悉的问题。可用启发式解决方案来解决这些问题。

距离衡量在各个维上将各自属性权值$w_1, w_2, \cdots, w_n$合并在一起考虑：

$$\sqrt{w_1^2(x_1-y_1)^2+w_2^2(x_2-y_2)^2+\cdots+w_n^2(x_n-y_n)^2}$$

当属性权值存在类特性的情形时，每个类各自拥有一组属性权值。

对每个训练实例进行分类之后，所有的属性权值都要被更新，更新是基于最类似的样本（或每个类最类似的样本）。训练实例x，最类似样本y，对每个属性i来说，差异$|x_i-y_i|$是这个属性对分类决定所做贡献的一种衡量。如果差异很小，这个属性的贡献是正向的，而差异很大时，贡献则是负向的。基本的理念是依据这个差异的大小以及分类是否确实正确，来更新第i个属性的权值。如果分类正确，相关的权值增加；如果分类错误，则减小。增加或减少的幅度由差异大小来控制，如果差异小，幅度便大，反之亦然。跟随权值改变之后的通常是重整化（renormalization）步骤。一个可能达到相同效果而更为简单的方法是，如果决策正确，权值不变；如果错误，则增加差异最大的那些属性的权值，以强调差异。Aha（1992年）对这些权值调整的算法进行了详细介绍。

一个检验属性加权方案是否有效的好方法是在数据集所有实例中添加无关属性。理想状况是，无关属性的介入，既不影响预测质量，也不影响储存样本的数目。

6.4.4 推广样本集

被推广了的样本集是实例空间上的矩形区域，因为它们是高维的，因此称为超矩形
238 (hyper-rectangles)。在对新的实例进行分类时，很有必要按下面所述的方法来修正距离函数，使之能计算实例与超矩形间的距离。当一个新实例被正确分类时，推广便是简单地将它与同一个类中最近邻的样本合并起来。最近邻样本不是单个的实例就是一个超矩形。在前一种情形下就产生了一个新的超矩形，它包含新的和旧的实例。在后一种情形下，超矩形就会扩大，将新的实例包含进去。最后，如果预测不正确，而且导致做出错误预测的是一个超矩形，那么这个超矩形的边界就会改变，从而退离新的实例。

很有必要在一开始就决定是否允许由于超矩形嵌套或重叠而引起的过度推广。如要避免过度推广，在推广一个新实例之前要进行检查，看看新的超矩形是否会与任何属性空间的区域相冲突。如果有冲突便放弃推广，将这个实例如实储存起来。注意超矩形的重叠正如同在一个规则集里，同一个实例被两条或两条以上的规则所包含。

在某些方案中，当一个经推广的样本集被完全包含在另一个中时，可使用嵌套，就如同在某些表达中，规则会有例外。为达这个目的，每当一个实例被错误分类时，应用一个后退启发式方法，即如果第二近邻样本能产生正确预测，使用第二近邻样本，再次试行推广。这种使用第二次推广机会的方法促使超矩形的嵌套。如果一个实例正好落入一个代表错误类的矩形中，而这个矩形中已含有一个与此实例同类的样本，它们两个便会进行推广，形成嵌套在原来的矩形中的一个新的“例外”超矩形。对于嵌套推广样本，为了防止同属一个类的所有实例推广成单个的矩形而占据问题空间的大部分，学习过程经常始于一小部分的实例。

6.4.5 用于推广样本集的距离函数

对于推广样本集，很有必要对距离函数进行推广，用于计算某个实例离开一个推广样本或另一个实例的距离。当实例点落在超矩形内时，这个实例离开超矩形的距离被定义为零。一个最为简单的用于计算位于超矩形外的实例离开超矩形距离的方法便是在超矩形内选择离开这个实例最近的实例、并计算它们之间的距离。但是，这种方法削弱了推广的益处，因为它引发了对某个具体实例的依赖。更精确地说，正好落入超矩形的新实例能继续保留推广的益处，而位于超矩形之外的则不能。或许使用（新实例）离开最为靠近的部分超矩形的距离来替
239 代会更好些。

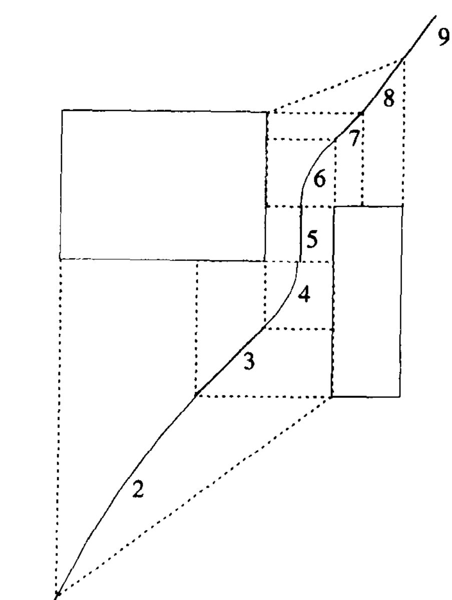


图6-14 两个矩形类之间的分界线

图6-14展示了使用最近邻点距离衡量方法时，两个矩形类之间的隐含边界。即使是在二维空间，边界也包含九个区域（图中用数字标出，方便辨认），那么在高维超矩形的情况下就更为复杂了。

从左下角开始，第一个区域分界线是呈线性的，区域位于两个矩形的延伸范围之外，即大矩形两边

界的左方，小矩形两边界的下方。第二个区域是在一个矩形的延伸范围之内，即大矩形左边界的右方，但在另一个矩形延伸范围之外，小矩形两边界的下方。在这个区域分界线呈抛物线，因为离一条给定线和一个给定点距离相等的所有点的集合是一条抛物线。第三个区域是 240
分界线向上能投射到大矩形的下边界而向右能投射到小矩形的左边界的区域。这条分界线呈线性，因为它离两条矩形边界是等距离的。第四个区域分界线位于大矩形的右下方。这时分界线是呈抛物线，因为它是离大矩形右下角和小矩形的左边界等距的点的集合。第五个区域位于两个矩形之间：这里分界线是垂直的。右上部分图形重复这样的模式：先是抛物线，接着是线性，然后又是抛物线（虽然这条抛物线和直线很难区分），最后是线性分界线，脱离了两个矩形范围。

这个简单的情形定义了一个复杂的分界边界！当然没有必要非常清楚地表达分界线，它是通过最近邻法计算得出的。然而这还不是很好的方法。用超矩形中的最近邻实例来计算距离过度依赖与某个具体位置上的实例，即过度依赖与矩形的某个角，最近邻实例可能离这个角较远。

最后来关注一下重叠或嵌套超矩形的距离衡量问题。这时，因一个实例可能落入多于一个的超矩形中，使问题复杂化了。适用于这种情况的一个启发式方法是选择包含这个实例且超矩形最为具体化的类，即覆盖实例空间最小的类。

无论是否允许重叠或嵌套，都要修正距离函数，以考虑样本预测正确率以及不同属性的相对重要性，正如前一节所述的修剪干扰样本和属性加权那样。

6.4.6 推广的距离函数

定义一个距离函数有很多方法，也很难找出理由来做任何一种具体的选择。一个较好的方法是考虑将一个实例通过一系列预设的初级运算转换成另一个实例，然后计算当运算方式为随机选择时这个系列运算会出现的概率。如果考虑所有可能的转换路径，用它们的概率进行加权，这将使方案更趋成熟。将这个方法自然推广到计算某个实例和其他一系列实例之间 241
的距离问题时，便是要考虑将这个实例转换为系列中的所有实例。通过这种技术，使得考虑每个实例“影响范围”成为可能，这个范围是软边界而非k最近邻规则对某个实例的决策不是“内”就是“外”的硬边界分隔。

用这种方法，对于一个类未知的测试实例，依次计算它和每个类的训练实例集的距离，选择距离最近的类。这种通过定义不同的转换集的基于转换的方法对名词性属性和数值属性都同样适合，甚至可用于考虑一些不常用的属性，如弧度、星期等，它们都是循环量度的。

6.4.7 讨论

继Aha（1992年）发现基于实例的学习方法结合修剪干扰样本和给属性加权，能达到较其他学习方法更好的工作效果之后，最近邻法在机器学习领域得到普及。值得一提的是虽然我们只是在分类中讨论这个问题而没有对数值预测进行讨论，但它们是同等适用的：可以由k个最近邻的预测值根据距离加权组合起来得出最终的预测。

从实例空间的角度看，标准规则和基于树的表示方法，只能表示与属性定义的轴相平行的类边界。这并不妨碍名词性的属性，但对数值属性有障碍。非轴平行的类边界只能根据在边界上下的一些轴平行矩形所覆盖的区域进行近似，矩形的数目决定了近似度。相反，基于

实例的方法可以很容易地表示任意的线性边界。即使是一个二类问题而每个类只有一个实例，由最近邻规则得到的边界是一条任意方向的直线，即两个实例连线的垂直平分线。

简单的基于实例的学习方法不提供清楚的知识表达，除了选择代表样本。但当它和样本推广结合在一起时，可得到一组规则集，这组规则集可用来与其他机器学习方案所产生的规则集相比较。这些规则更趋于稳定，因为经修改的、结合了推广样本的距离函数，可用于处理不落入规则内的实例。这减少了规则要覆盖整个实例空间，甚至所有训练实例空间所带来的压力。而另一方面，大多数基于实例的学习方案的递增特性，意味着在只看到一部分训练集时，便很迫切地产生规则，而这不可避免地降低了规则质量。

因为不很清楚最好用何种方法进行推广，我们没有提供各种运用推广的基于实例学习的
242 精确算法。Salzberg（1991年）提出使用嵌套样本的推广可以在一些不同分类问题中取得较高的分类正确率。Wettschereck 和 Dietterich（1995年）辩论说这些结论是偶然的，在其他领域并不成立。Martin（1995年）揭示了性能表现差的原因是因为当超矩形重叠或嵌套而引起过度推广，而非推广造成性能差。他还证实了在很多领域中，如果避免了重叠或嵌套的发生，会得到很好的效果。Cleary 和Trigg（1995年）提出了基于转换的推广距离函数。

样本推广是一种少见的学习策略的例子，它的搜寻过程是从具体到一般，而不像树或规则归纳过程是从一般到具体。没有特别的理由能使人信服这个从具体到一般的搜寻必须对实例采用严格的递增模式考虑，使用基本的基于实例的方法来产生规则的批处理方案也是存在的。另外，进行保守的推广，对没有覆盖的实例挑选“最近”推广是非常好的主意，最终可以延伸为普通树和规则的归纳。

6.5 数值预测

用于数值预测的树就像普通的决策树，但在叶节点所储存的是代表叶节点处实例平均值的一个类值，这种树称为回归树，或者是在叶节点储存了能预测达到叶节点的实例类值的一个线性回归模型，这种树称为模型树。下面我们要讨论模型树，因为回归树实际上是一种特殊情况。

回归树和模型树先是使用决策树归纳算法建立一个初始树。但是，在大多数的决策树算法中，属性分裂是根据信息增益最大值来选择的，那么在数值预测中，适合使每个分支子集内部类值变化最小化。一旦形成了基本树，就要考虑从每个叶节点往回修剪树，就像普通决策树一样处理。回归树和模型树归纳法的区别只是在于，后者的每个节点用一个回归平面来替代一个常量值。参与准确定义回归的属性，正是那些参与决定子树修剪的属性，即在当前
243 节点的下层节点中。

在广泛讨论模型树之后，将简单地讨论怎样从模型树中产生规则，然后介绍另一种数值预测方法、局部加权线性回归。模型树是从基本的分治决策树算法演绎而来，局部加权线性回归则是受启于前一节讨论的基于实例的分类方法。如同基于实例的学习，它是在预测阶段进行所有“学习”的。局部加权线性回归法利用线性回归使模型局部拟合与具体的实例空间来组建模型树，但它采用了相当不同的方法。

6.5.1 模型树

当使用模型树来对一个测试实例进行数值预测时，像普通的决策树一样，在每个节点根

据实例的属性值来决定走向，直至树的叶节点。叶节点含有一个基于部分属性值的线性模型，使测试实例得到一个原始的预测值。

在修剪过的树的两个相邻叶节点的线性模型之间会不可避免地产生突变，使用平滑处理补偿这个突变来替代直接使用原始的预测值会更加有益。这是在小规模的训练集上构建模型的特别问题。平滑可以通过在建树的时候，如同在叶节点一样，为每个内部节点建线性模型来实现。当一个测试实例根据叶节点模型得到一个原始预测值时，这个值沿着树一路过滤返回根节点，在每个节点将这个值与该节点的线性模型所提供的预测值结合进行平滑处理。

一个适当的平滑计算是

$$p' = \frac{np + kq}{n + k}$$

这里p'是要向上一层节点传输的预测值，p是由下层节点传输上来的预测值，q是这个节点提供的预测值，n是下层节点的训练实例数量，k是平滑常量。试验证明平滑处理大大提高了预测的正确性。

在树建完后再把内部节点模型组合到叶节点模型中，可以达到同样的平滑处理效果。那么在分类过程中，只需要用叶节点模型。缺点是叶节点模型变得大而难以理解，因为当内部节点模型加入后，很多原来为零的系数现在变为非零系数了。 244

6.5.2 建树

分裂标准用于决定对某个具体节点的训练数据T，按哪个属性分裂最好。基于把数据T中类值的标准差看作是对这个节点的误差衡量，并且计算期望误差减少值作为对这个节点每个属性进行测试的结果。选择使期望误差减少值达到最大的属性作为这个节点的分裂属性。

期望误差减少值称为SDR，即标准差减少值（standard deviation reduction），计算如下：

$$\text{SDR} = sd(T) - \sum_i \frac{|T_i|}{|T|} \times sd(T_i)$$

这里$T_1, T_2, \cdots, T_i$是根据所选属性在节点进行分裂的结果数据集。

当在一个节点的实例类值变化非常细微时，便终止分裂过程，即当标准差只是占原始标准差的一小部分时（比如小于5%）。当只剩下很少的实例时，比如4个或更少，也终止分裂。试验证明所得的预测结果对这些阈值选择并不很敏感。

6.5.3 修剪树

如前所述，不仅在树的每个叶节点有一个线性模型，在每个内部节点上也要有，这是为了进行平滑处理的需要。在修剪之前，未修剪树的每个节点上都有一个模型。模型的形式为

$$w_0 + w_1 a_1 + w_2 a_2 + \cdots + w_k a_k$$

这里$a_1, a_2, \cdots, a_k$是属性值，权值$w_1, w_2, \cdots, w_k$用标准回归法计算。然而，只有这个节点下层子树的测试属性才用于回归，因为其他影响预测值的属性已在引入这个节点的测试中考虑进去了。注意，前提是我们假设它们都是数值属性：在下一节中讨论名词性属性的处理方法。

修剪过程使用了一个估计器，是在每个节点、对测试数据期望错误的估计器。首先，将这个节点上所有训练集实例的预测值和真实类值之间的绝对偏差进行平均值计算。由于树是

用这个数据集建立的，这个平均值对于未见情形来说，是个低估的期望误差。为了补偿这一点，将它与系数$(n+v)/(n-v)$相乘，这里n是这个节点的训练实例数量，v是给出这个节点预测
245 类值的线性模型所用的参数数量。

在某个节点对测试数据的期望误差计算如上所述，使用线性模型进行预测。因为有补偿系数$(n+v)/(n-v)$，可以通过减少项数使估计误差达到最小化，从而使线性模型进一步简化。减少一项便减小了相乘系数，这也许足以平衡在训练实例上平均误差的不可避免的增加。只要估计误差还在降低，便继续采用贪心式逐个减少项数。

最后，一旦每个内部节点的线性模型都已到位，只要期望估计误差还在降低，便从叶节点返回修剪树。将节点的线性模型期望误差与其子树的模型期望误差进行比较。为了计算后者，将来自每个分支的误差组合起来产生一个综合值。这个综合值是根据分支训练实例的数量比率对分支加权，利用这些权值将误差估计进行线性组合。

6.5.4 名词性属性

在建一个模型树之前，所有的名词性属性都被转换成二进制变量，随后便被当作数字一样对待。对每个名词性属性，根据训练实例计算每个可能的属性枚举值所对应的平均类值，然后按照这些平均值将属性枚举值按序排列。如果名词性属性有k个可能的属性值，就要用$k-1$个合成二进制属性来替代。如果属性值是属性序列中前i个中的一个，那么第i个属性值为0，否则为1。这样所有的分裂都是二分的：所牵涉到的不是一个数值属性就是一个可与数值属性同样对待的合成二进制属性。

可以证明在某个节点对于含k个值的名词性变量，最好的分裂点是按每个属性值的平均类值大小排序所得到的$k-1$个位置中的一个。这个排序过程确实要在每个节点重复进行；但是，由于在树的下层节点中训练实例数目很小（在某些情况下，节点没有表现出某些属性所有的值），难免增加了干扰，而只在建模型树之前进行一次排序并不损失很多。

6.5.5 残缺值

为了考虑残缺值，对标准差减少值（SDR）公式进行了修改。包括残缺值补偿，最后的公式为

246

$$\mathrm{SDR}=\frac{m}{|T|}\times\left[sd(T)-\sum_{j\in\{L,R\}}\frac{|T_j|}{|T|}\times sd(T_j)\right]$$

这里m是属性没有残缺值的实例数量，T是在这个节点上的实例集，T_L，T_R是用这个属性进行分裂所得的两个实例集，对属性的所有测试都是二分的。

在处理训练和测试实例时，一旦某个属性被选定用于分裂就必须将实例根据它们各自在这个属性上的数值分成两个子集。很明显，当属性值残缺时就会发生问题。一种有趣的技术称为代理分裂（surrogate splitting）用于处理这类问题。它要寻找另一个分裂属性来代替原来的分裂属性。被选择的属性与原来的属性相关性最高。但是，这种技术不仅复杂而且执行起来很耗时。

一个比较简单的方法是用类值作为代理属性，相信根据推理，它是最有可能与分裂属性相关的一个属性。当然这只能是用于处理训练实例，因为测试实例的类是未知的。解决

测试实例的一个简单方法是用这个节点上的训练实例的对应属性的平均值来代替这个未知的属性值，对二值属性来说，结果是选择拥有多数实例的子节点。这个简单的法在实际应用中效果不错。

现在来仔细看一下怎样在训练过程中利用类值作为代理属性。首先处理分裂属性值已知的所有实例。用常规方法来决定分裂阈值，将实例按属性值排列，对每个可能的分裂点根据上述公式计算SDR，选择误差减少值最大的分裂点。只有分裂属性值已知的实例才参与决定分裂点。

然后根据测试将这些实例分成L和R两个子集。再来决定L或R中的实例哪个平均类值较大，并计算这两个平均类值的平均值。这样，属性值未知的实例就根据它的类值是否超过总体平均值来决定将它放入L还是R中。如果超过总体平均值，它将加入L或R中具有较大平均类值的那个，否则便加入具有较小平均类值的那个。当分裂停止时，所有的残缺属性值都用叶节点上的训练实例的相应属性平均值来替代。

6.5.6 模型树归纳伪代码

图6-15给出了模型树算法的伪代码。其中有两个主要部分，一个是通过连续不断地分裂节点来建树，由split来实现，还有一个是从叶节点向上修剪树，由prune来实现。节点(node)的数据结构包括：标明这个节点是内部节点还是叶节点的类型标记；指向左分支和指向右分支的指针；到达这个节点的实例集；这个节点的分裂属性；以及一个代表这个节点线性模型的结构。 247

```
MakeModelTree (instances)
{
  SD = sd(instances)
  for each k-valued nominal attribute
    convert into k-1 synthetic binary attributes
  root = newNode
  root.instances = instances
  split(root)
  prune(root)
  printTree(root)
}
split(node)
{
  if sizeof(node.instances) < 4 or sd(node.instances) < 0.05*SD
    node.type = LEAF
  else
    node.type = INTERIOR
    for each attribute
      for all possible split positions of the attribute
        calculate the attribute's SDR
    node.attribute = attribute with maximum SDR
    split(node.left)
    split(node.right)
}
prune(node)
{
  if node = INTERIOR then
```

图6-15 模型树归纳伪代码

```
    prune(node.leftChild)
    prune(node.rightChild)
    node.model = linearRegression(node)
    if subtreeError(node) > error(node) then
      node.type = LEAF
}
subtreeError(node)
{
  l = node.left; r = node.right
  if node = INTERIOR then
    return (sizeof(l.instances)*subtreeError(l)
          + sizeof(r.instances)*subtreeError(r))/sizeof(node.instances)
  else return error(node)
}
```

248 图 6-15（续）

主程序一开始就调用sd函数，在split一开始又再次调用，用于计算一个实例集类值的标准差。接着是如前所述的转换合成二进制属性过程。创建一个新节点以及输出最终树的标准程序没有在这里展示。在split中，sizeof返回一个集合中所含元素数量。残缺属性值的处理采用前面所述的方法。SDR根据上一子节开始的公式计算。虽然没有在代码中显示，但如果根据一个属性分裂所产生的叶节点包含实例数少于两个，它的结果便被设定为无穷大。在prune中，linearRegression程序沿着子树一直向下搜集属性进行递归，对节点所含实例的这些属性执行线性回归形成函数，然后如前所述只要能改善误差估计，贪心式地减少项。最后，误差error函数返回

$$\frac{n+v}{n-v}\times\frac{\sum_{\text{实例}}|\text{预测类值的偏差}|}{n}$$

这里n是节点上的实例数量，v是节点线性模型的参数数量。

图6-16是利用这个算法对一个含有两个数值属性和两个名词性属性问题建立模型树的例

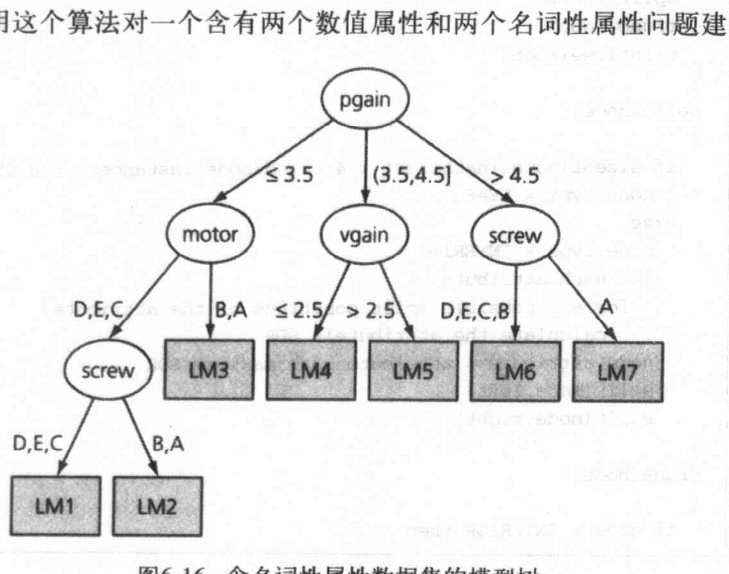


图6-16 含名词性属性数据集的模型树

子。要预测的是模拟伺服系统的上升时间，系统包括伺服放大器、电动机（motor）、导螺杆（lead screw）和滑架。名词性属性在其中担当着重要的角色。各含有5个属性值的名词性属性motor和screw由4个合成二进制属性所替代，表6-1列出了对应的两组值。这些属性值的顺序：motor为D、E、C、B、A，screw正巧也是D、E、C、B、A，都是由训练数据决定的。使用motor = D的所有实例的上升时间平均值小于使用motor = E的所有实例的上升时间平均值，使用motor = E 的又小于使用motor = C的，依次类推。从表6-1所列的系数大小可以明显看出，motor = D对于 E、C、B、A在LM2模型中起着主导作用；motor = D、E对于C、B、A在LM1模型中起着主导作用。motor 和screw在某些模型中都处于次要角色。决策树展示了对数值属性实行三分的方法。起先是按常规建立二分树，但出现了根节点和它的一个子节点使用相同的测试属性pgain，因此用了一个简单的算法将这两个节点合并成如图所示的更易理解的树。

表6-1 模型树中的线性模型

模 型		LM1	LM2	LM3	LM4	LM5	LM6	LM7
Constant term		-0.44	2.60	3.50	0.18	0.52	0.36	0.23
pgain								
vgain		0.82		0.42				0.06
motor = D	vs. E, C, B, A		3.30		0.24	0.42		
motor = D, E	vs. C, B, A	1.80			-0.16		0.15	0.22
motor = D, E, C	vs. B, A				0.10	0.09		0.07
motor = D, E, C, B	vs. A			0.18				
screw = D	vs. E, C, B, A							
screw = D, E	vs. C, B, A	0.47						
screw = D, E, C	vs. B, A	0.63		0.28	0.34			
screw = D, E, C, B	vs. A			0.90	0.16	0.14		

6.5.7 从模型树到规则

模型树从本质上来看就是在叶节点带有模型的决策树。同决策树一样，它们都会存在 249~250
3.3节中提到的子树复制的问题，有时用规则集来代替树表示这个结构会更加精确一些。可以从数值预测中产生规则吗？回顾一下6.2节中所述的规则学习器，它是用割治法同局部决策树共同协作来从树中提炼决策规则的。可以应用同样的策略从模型树中产生用于数值预测的决策列。

先用所有的数据建一个局部模型树。选择其中的一个叶节点把它转变为一条规则。将这个叶节点所涵盖的数据去除，重复上述步骤对剩余的数据进行处理。问题是怎样来建局部模型树，即一个未展开节点的树？这个问题归结为怎样选择下一个展开节点。图6-5（6.2节）的算法选择了类属性的熵最小的那个节点。对预测结果是数值的模型树来说，只要简单地用方差来替代。这是基于相同的推理：方差越小，子树越浅，规则越短。算法的其余部分都维持不变，用模型树学习器的分裂选择法和修剪策略代替决策树学习器的分裂和修剪。因模型树的叶节点是线性模型，相应地，规则的右边也将是线性模型。

如此使用模型树产生规则集有一点要加以说明：就是模型树学习器所用的平滑处理。使用经平滑处理的模型树不能降低最终的规则集预测的误差。这也许是因为平滑处理最好是用于连续数据，但在割治法中前一条规则所覆盖的数据被去除了，在数据分布上留下了空洞。如果平滑要进行的话，必须在规则集产生后实行。

6.5.8 局部加权线性回归

数值预测的另一种方法就是局部加权线性回归法。在模型树中，树结构将实例空间分隔成不同区域，每个区域中都有一个线性模型。实际上是训练数据决定了实例空间如何分区。局部加权回归却是在预测时间产生局部模型的，它是通过赋予测试实例的近邻实例较高的权值来实现的。具体地说，对训练实例根据它们离测试实例的距离来加权，然后在加了权的数据上进行线性回归。靠近测试实例的训练实例权值较高，远离测试实例的权值较低。换句话
251 说，是为某个具体的测试实例特制一个线性模型，并用它预测这个测试实例的类值。

为了要使用局部加权回归，必须为训练实例决定一个基于距离的加权方案。一种常规选择是根据训练实例离开测试实例的欧几里得距离的倒数来进行加权。另一种可能的选择是利用欧几里得距离和高斯核函数协作来决定加权。然而，没有证据证明加权方案的选择是关键。更重要的是用于衡量距离函数的“平滑参数”（smoothing parameter）的选择，距离要和这个参数的倒数相乘。如果这个值设得小，只有非常靠近测试实例的实例才会得到显著的权值；如果这个值设得大，则更多远距离的实例对模型的建立也会有着显著影响。选择平滑参数的一种方法是将其设定为离开第k个最近邻训练实例的距离，从而随着训练实例数量增加，这个值会越来越小。k值的最佳选择依赖于数据中的干扰多少。干扰实例存在越多，越多的近邻实例需要被包含在线性模型中。通常，合适的平滑参数是通过交叉验证得到的。

像模型树一样，局部加权线性回归法可以接近非线性函数。它的一个主要优点就是非常适合于递增学习：所有训练都在预测时间完成，因此新的实例可以随时加入到训练集中。但正如其他基于实例的学习一样，获得对一个测试实例的预测会很慢。首先，要扫描训练实例计算它们的权值，然后再对这些实例实行加权线性回归。另外，像其他基于实例的方法一样，局部加权回归几乎不能提供有关训练数据集全局结构的信息。注意，如果平滑参数是基于第k个近邻实例的，并且加权函数赋予远距离实例的权值为零，使用4.7节中讨论的kD树和球树可以令寻找相关近邻实例的速度提高。

局部加权学习并不只局限于线性回归，它可应用于任何能够处理加权实例的学习技术。特别是可以应用于分类。大多数的算法都很易调整以适应处理权值。诀窍是能意识到（整数）权值可以模拟成创建同一实例的几个拷贝。每当学习算法计算模型使用一个实例时，就假设这个实例同时有恰当数量的相同实例伴随着。如果权值不是整数，这点也同样适用。比如在4.2节中讨论的朴素贝叶斯算法乘以源自实例权值的合计数。瞧，你拥有了一个可以用于局部加权学习的朴素贝叶斯法版本。

实践证明局部加权的朴素贝叶斯法工作非常出色，比朴素贝叶斯法本身以及k最近邻技术都要好。它放宽了朴素贝叶斯固有的独立假设，与更为复杂的增强朴素贝叶斯法进行比较，
252 也能获得较好的比较结果。局部加权学习的独立假设仅仅是在近邻实例之间，而不像标准的朴素贝叶斯是在整个实例空间的全局独立假设。

原则上，局部加权学习也可以应用到决策树以及其他一些比线性回归和朴素贝叶斯更为复杂的模型。但是，应用局部加权学习获益较少，因为它从根本上来说是一种能让简单模型更具灵活性的方法，是通过允许简单模型接近任意目标来实现的。如果基本学习算法已经能做到这点，便没有什么理由应用局部加权学习了。尽管如此，它仍可以改善其他一些简单模型，如线性支持向量机和logistic回归模型。

6.5.9 讨论

回归树是由Breiman等（1984年）在CART系统中介绍的。CART是“分类和回归树”(classification and regression trees)，用于离散类的决策树归纳，这和C4.5很相像，C4.5是单独创建的，作为用于归纳回归树的一种方案。上述许多技术，如处理名词性属性的方法和处理残缺值的代理机构，在CART中都包含了。模型树只是在近期才出现的，最初是由Quinlan（1992年）提出的。利用模型树来生成规则集（虽然不是局部树），是由Hall等（1999年）探索的。

模型树归纳并不像决策树归纳那样通用，部分是因为易于理解的描述（以及实施）技术只是在近年才有的（Wang 和 Witten，1997年）。对数值预测，神经网络更为常用，虽然神经网络存在着缺点，它所生成的模型结构不明朗以及不能帮助理解方案的本质所在。即便现在已有能提供可理解的、洞察神经网络结构的技术，内部表达的任意性意味着由相同数据训练出来的相同网络体系也许存在着戏剧性的变异。将所归纳的函数分隔成线性模块，模型树提供了可复制、至少较易理解的一种表示法。

有很多不同的局部加权学习方法。例如，统计学家们考虑用局部二次模型代替线性模型，应用局部加权logistic 回归来处理分类问题。还有可以在文献中找到许多潜在的不同的加权和距离函数。Atkeson 等（1997年）写出了一篇关于局部加权学习非常出色的调研报告，主要是回归问题。Frank 等（2003年）对局部加权学习和朴素贝叶斯法结合应用进行了评估。 253

6.6 聚类

在4.8节中我们考察了k均值聚类算法，它是选择k个初始点代表k个初始的聚类中心，所有的数据点都被赋予最靠近的聚类中心，计算每个聚类中数据点的平均值作为新的聚类中心，如此不断重复直至聚类不再变化。这个程序只有当事先知道聚类数目才可行，这节以讨论聚类数目未知时该如何处理作为开始。

接着考察两种不像k均值那样把实例分成不相交聚类的技术。第一种是在20世纪80年代末期创建的包含在Cobweb（用于名词性属性）和Classit（用于数值属性） 两个系统中的递增聚类方法。这两个系统都是用等级分组实例，衡量的是聚类的“质量”，称为类别效用(category utility)。第二种是一种统计学聚类方法，它是基于不同概率分布的一种混合模型，每个聚类有一个概率分布。它对实例所属类的赋予是概率性的，而不是决定性的。我们将要解释一些基本技术，然后介绍一种易于理解的、称为AutoClass的聚类方案是如何工作的。

6.6.1 选择聚类的个数

假设要使用k均值法，但聚类的个数未知。一种解决方法是对不同的可能个数进行试验，看看哪个最好，即哪个能使所有点离开它们聚类中心的距离平方的总和达到最小。一个简单的方法是从一个给定的最小个数开始，或许是$k = 1$，然后一直试验到一个较小的、固定的最大值，用交叉验证法找出最好的个数。由于k均值法很慢，使用交叉验证使之更慢，因此试图对许多可能的k值进行试验不太可行。注意，根据距离平方总和标准来决定“最佳”聚类训练数据的方法，将总是选择和数据点一样多的聚类！为了抑制选择很多聚类的方案，必须应用诸如在5.10节中所述的最短描述长度准则，或采用交叉验证。

另一种可能方法是开始先找出很少几个聚类，然后决定是否值得再将它们分裂。你可选

择k=2、执行k均值聚类直到它终止，然后考虑分裂每个聚类。如果最初考虑的二分聚类是不
254 能取消的，并且每个部分的分裂是独立考察的，那么计算时间将大大减少。分裂聚类的一种方法是在变化最大的方向、距离聚类中心一个标准差的位置产生一个新的种子，随后在反方向、等距建立第二个种子。（如果速度太慢，另一种方法是在任意方向，选择一个聚类边界盒的距离比例项。）然后运用这两个新的种子，对聚类中的点进行k均值聚类。

聚类分裂暂时完成，是否值得保留分裂，或者原来的聚类也是合理的？察看所有点离开聚类中心距离平方总和是没有用的，两个子聚类的总和一定是较小的。创建一个额外的聚类，会招致惩罚，这是最短描述长度准则的工作范畴。利用这个原理来察看详细说明两个新聚类中心以及每个点与它们之间关系所需的信息，是否超过详细说明原先的聚类中心以及所有点与它们之间关系所需要的信息。如果是超过了，那么新的聚类是徒劳的，将被放弃。

如果分裂被保留下来，试着对每个新的聚类进一步分裂。这个过程一直继续到不再有值得保留的分裂。

将这个迭代聚类过程和4.8节中提出的kD树或球树数据结构结合起来，可在实现过程中获得额外的效率。那样，是从根节点沿着树一直向下得到数据点。当考虑分裂一个聚类时，没有必要考虑整个树，只需考虑这个聚类所覆盖的部分。例如，当决定是否分裂图4-16a左下方（粗线下方）的聚类时，只需考虑图4-16b中的树节点A和B，因为节点C 与这个聚类无关。

6.6.2 递增聚类

k均值算法对整个数据集进行迭代运算直至收敛，而下面考察的聚类方法则是递增型工作的，它是一个实例接一个实例进行的。在任何阶段，聚类在叶节点包含实例形成一个树结构，根节点代表着整个数据集。开始时树只有一个根节点。实例一个个加进来，树则在每个阶段进行适当的更新。更新也许只是寻找恰当的位置来放置代表新实例的叶节点，或者是要重建受到新实例影响的部分树。决定怎样更新以及在哪里更新的关键是称为类别效用的量，它衡量将实例集划分成聚类的总体质量。将在下一节中详细讨论它是怎样定义的。先来看一下聚
255 类算法是如何工作的。

这个过程最好用一个例子来说明。再次使用大家熟悉的天气数据，但不包括属性玩。为了便于跟踪程序，14个实例分别被标为$a, b, c, \cdots, n$（如表4-6）。出于兴趣，我们包含了类标yes或 no，尽管必须强调对于这个人造数据集来说，假设实例的两个类必须是两个完全分隔的类别范畴几乎是没有什么理由。图6-17展示了聚类过程中出现的一些重要情形。

一开始，当新的实例被纳入结构中时，它们各自形成顶层总聚类下的子聚类。每个新实例都被试探着放入现有的叶节点，然后对顶层节点的子节点集进行类别效用评估，看看这个叶节点是否是新实例的一个好的“接受体”(host)。对前五个实例来说，没有这样的接受体：最好根据类别效用让每个实例形成一个新的叶节点。到第六个实例终于可以形成一个聚类了，将新实例f和旧实例（即接受体）e结合在一起。再回过头来看表4-6（4.3节），你将发现第五个和第六个实例的确非常相似，只有刮风属性值不同（还有这里忽略的玩属性值不同）。接下来一个实例g被置入同一个聚类中（它和e比较，只是阴晴属性值不同）。随之而来的是再次聚类运行过程。首先，对g进行评估，看看根节点的五个子节点中哪个能成为最佳接受体；结果是已经成为聚类的最右边的那个子节点。然后运用聚类算法将这个节点作为根节点，对它的两个子节点进行评估，看哪个是较好的接受体。在这个例子中，根据类别效用衡量结果，增加新实例作为一个子聚类是最好的。

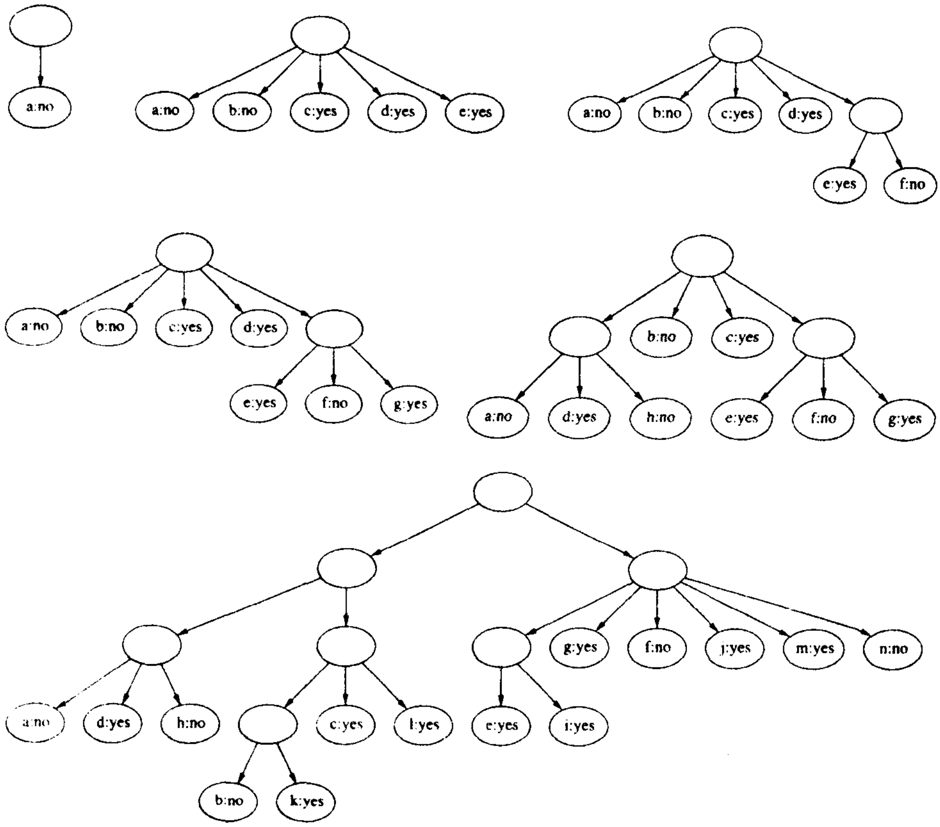


图6-17 对天气数据的聚类

如果按这样的方法继续下去，不会有从根本上重建树的机会，那么最终的聚类结果会过度依赖于实例的排序。为了避免这点，有一些重建的规则，在图6-17中展示的下一步，当实例*h*加入进来时，你便能看到。这时，两个现存的节点合并成一个聚类：在新实例*h*添加进来之前，节点*a*和*d*被合并。一种实现的方法是考虑所有成对节点的合并，评估每对的类别效用。然而，计算耗费很大，如果在每个新实例添加时都运行的话，会带来许多重复工作。

作为替代，另一种方法是每当为寻找合适的接受体而对某层的节点进行扫描时，同时记录下最适合的节点（能对这层的分裂产生最大类别效用的节点）和第二适合节点。最好的那个作为新实例的接受体（除非新实例自身作为一个聚类会更好）。但在将新实例加入接受体之前，先考虑接受体和第二适合节点的合并。在本例中，*a*是首选接受体而*d*是第二适合节点。对*a*和*d*的合并进行评估，结果是合并可以提高类别效用。因此这两个节点合并起来，产生在*h*被加入之前的图6-17中第五个分层结构。然后，考虑将*h*放置在新的、经合并的节点，最好的结果是如图所示将它作为一个子聚类。

与合并相反的操作称为分裂，也被实施，虽然在这个例子中没有涉及。每当鉴定出最好的接受体，而合并又证明无益处时，便考虑对接受体节点的分裂。分裂的实现正好与合并相 256~257
反，用子节点来替代这个节点。例如，要分裂图6-17中第四个分层结构的最右侧的节点，便将叶节点*e*、*f*和*g*提升一层，使它们与*a*、*b*、*c*和*d*成为同胞节点。合并和分裂为弥补由于不适当的实例次序所引起的错误选择，提供了一种递增重建树的方法。

14个实例最终的分层结构如图6-17最后所示。有两个主要的聚类，每个下面还有子聚类。如果play和don't play两种特性确实代表了数据的内在特性，那么期望的结果是每种各有一个聚类。从图中看不出这样清晰的结构，虽然（非常）有雅量的眼力通过将底层标有yes的实例组合在一起，同样将标有no的实例组合在一起，或许能辨别出一些细微倾向。对聚类做些仔细的分析会揭示出一些异常情况。（如果你要仔细分析，表4-6会有所帮助。）例如，实例*a*和*b*实际上是非常相像的，而它们却处在树的两个完全不同的部分。实例*b*最终和与它的相似度不如*a*的实例*k*聚在一起。实例*a*最终同实例*d*和*h*聚在一起，而*a*与*d*的相似度完全不及*a*与*b*的相似度。实例*a*和*b*被分开的原因是如前所述*a*和*d*合并了，理由是它们分别是实例*h*的最佳和次佳接受体。不幸的是*a*和*b*是最先的两个实例：如果两个中的任何一个再晚一点出现，也许它们最终就会聚在一起了。后续的分裂和重新合并也许能矫正这类异常，但在这个例子中却没有能矫正。

对数值属性可采用完全相同的方法。类别效用也同样对数值属性有所定义，基于对属性的平均值和标准差的估计。详细内容在下节中论述。然而有一个问题必须在这里提出：当估计某个节点的某个属性的标准差时，如果这个节点只包含一个实例，结果为零，只含一个实例的情况是较常出现的。不幸的是零方差在类别效用公式里会产生无穷大。一个简单的启发式解决方案是给每个属性强加一个最小方差。由于没有一种衡量法是绝对精确的，强加一个最小值还是合理的：它代表了对一个样本的测量误差。这个参数称为灵敏度（acuity）。

图6-18a在上部分展示了对部分鸢尾花数据集（30个实例，每个类各有10个实例）采用递增算法聚类的分层结构。在顶层有两个聚类（即代表整个数据集的节点的子聚类）。第一个包括鸢尾花 virginica 和鸢尾花 versicolor, 第二个只包含鸢尾花 setosa。鸢尾花 setosa本身又分裂成两个子聚类，其中一个含4个品种而另一个含6个。另一个顶层聚类分裂成三个子聚类，每个都含有相当复杂的结构。第一、二个都只包含鸢尾花 versicolor，除了一个例外，即每个都
258 含有一个离群的鸢尾花virginica；第三个子聚类只含 virginica 。这是对鸢尾花数据相当令人满意的聚类结果：它显示了这三个种类并不是人为的，而是反映在数据上存在真实的差异。但这还是一个有点过于乐观的结论，因为为得到这个适当的分类，必须对灵敏参数的设定做相当多的试验。

以此方案聚类，使每个实例产生一个叶节点。这使得合理大小的数据集不可抗拒地会形成一个很大的分层结构。从某种意义上说，即相当于对数据集的过度拟合。因此第二个参数称为截止（cutoff ）参数，用来限制结构增长。某些实例被断定与其他实例足够相似的则不准有它们自己的子节点，这个参数就是来掌控这个相似度阈值。截止参数是根据类别效用来制定，当加入一个新的节点所带来的类别效用增值达到足够小时，就将这个节点截掉。

图6-18b展示的同样是鸢尾花数据，但应用了截止参数聚类。许多叶节点含多个实例，它们的父节点被截掉了。由于一些详情被抑制了，三种鸢尾花的分区从这个结构图中就比较容易看了。同样，为得到这个聚类结果，必须对截止参数的设定做一些试验，而且事实上截止越是强烈，将导致越不能令人满意的聚类。

如果使用含150个实例的完整鸢尾花数据集，得到的聚类结果是相似的。但是，聚类结果还是有赖于实例的次序，图6-18是变更了输入文件的三种鸢尾花的次序所得到的结果。如果所有鸢尾花setosa第一个出现，接着是所有的鸢尾花versicolor和所有的鸢尾花 virginica，聚类结果则是相当差的。

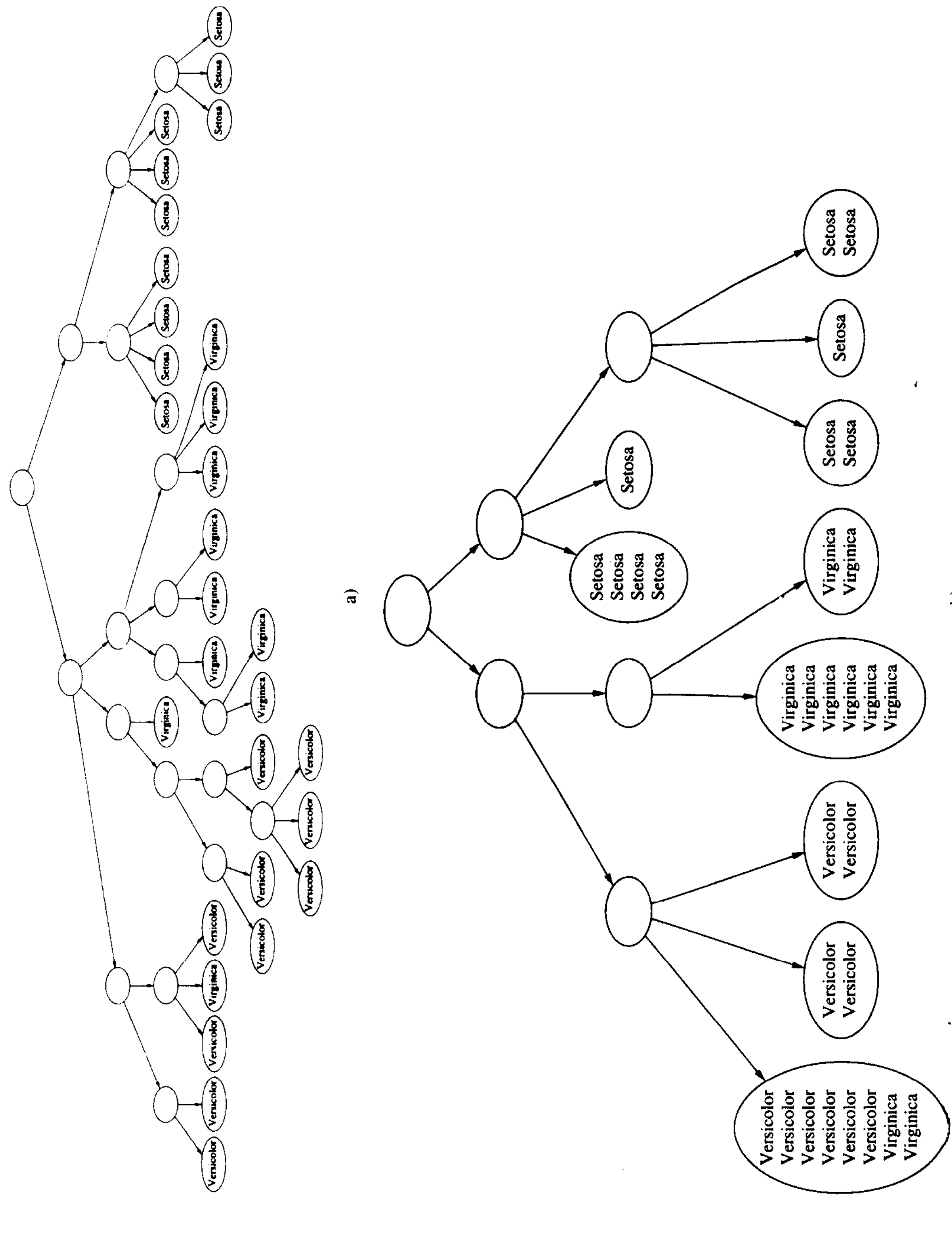


图6-18 鸢尾花数据的聚类层次结构

6.6.3 类别效用

现在来看怎样计算类别效用，它衡量将实例集分隔成聚类的总体质量。在5.9节我们从理论上学习了怎样应用最短描述长度原理来衡量聚类的质量。类别效用不是以MDL为基础的，而是一个类似于定义在条件概率上的二次损失函数。

类别效用的定义看起来是相当令人恐怖的：

$$CU(C_1, C_2, \cdots, C_k) = \frac{\sum_l \Pr[C_l] \sum_i \sum_j \left(\Pr[a_i = v_{ij} \mid C_l]^2 - \Pr[a_i = v_{ij}]^2 \right)}{k}$$

这里C_1、C_2，…，C_k 是k个聚类；外层的总和（运算）是针对这些聚类的；接下来的那个内层总和是针对属性的总和；a_i 代表第i个属性，它的值有v_{i1}，v_{i2}，…，v_{ij}，一共要处理j个。注意概率本身是总和所有实例得到的：因此还要内含一层总和。

259~260 如果多花点时间分析这个表达式，便能深入理解它的意义。聚类的意义在于它有利于更好预测聚类中实例的属性值，即在聚类 C_l 中的某个实例，属性a_i 值为v_{ij} 的概率$\Pr[a_i = v_{ij} \mid C_l]$，相对于概率$\Pr[a_i = v_{ij}]$来说，应是更好的估计，因为它考虑实例所在的聚类。如果不能从这个信息中获得帮助的话，则说明聚类并不理想。因此上述表达式多层总和的内部所计算的就是这个信息所能提供的帮助量，它是根据这两个概率平方的差值来衡量的。这并不是非常标准的差平方（squared-difference）的衡量方法，因为那是总和差值的平方（能产生对称结果），而现在的计算是总和平方的差值（虽然适合但不是对称的）。里面的两层总和符对所有属性、所有可能属性值的概率平方的差进行总和计算。然后外面的那个总和符对所有的聚类，利用它们各自的概率进行加权，进行总和计算。

最后总数除以k有点难以说明理由，是因为已经总和所有类的平方差。类别效用提供"每个聚类"的指标数来阻止过度拟合。否则的话，由于概率是由总和适当的实例获得的，如每个实例都能处于合适自己的聚类，将获得非常好的类别效用。那么，对属性a_i 来说，当属性值为聚类 C_l 中单个实例实际的a_i 属性值时，$\Pr[a_i = v_{ij} \mid C_l]$为1，而对于其他属性值这个概率都为0，类别效用计算公式的分子最终变为

$$n - \sum_i \sum_j \Pr[a_i = v_{ij}]^2$$

这里n是属性的总个数。这是分子所能达到的最大值，如果在类别效用计算公式中不除以k，便没有理由生成含有一个以上成员的聚类。把这个系数看成基本的过度拟合避免措施是最好不过了。

这个类别效用公式只适合于名词性属性。然而，假设属性是正态分布的，已给出（观察所得）平均值μ和标准差σ，很容易将其扩展应用于数值属性。属性a的概率密度函数为

$$f(a) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(a-\mu)^2}{2\sigma^2} \right)$$

属性值概率的平方总和计算可模拟为

$$\sum_j \Pr[a_i = v_{ij}]^2 \Leftrightarrow \int f(a_i)^2 \mathrm{d}a_i = \frac{1}{2\sqrt{\pi}\sigma_i}$$

这里σ_i 是属性a_i 的标准差。因此对数值属性，我们从数据中来估计标准差，所用数据既包含
261 某个聚类中的数据（得到σ_{il}）又包含所有聚类中的数据（得到σ_i），将这些运用到类别效用公

式中：

$$CU(C_1,C_2,\cdots C_k)=\frac{1}{k}\sum_l \Pr[C_l]\frac{1}{2\sqrt{\pi}}\sum_i\left(\frac{1}{\sigma_{il}}-\frac{1}{\sigma_i}\right)$$

这时，以前提到的当标准差估计为零时出现的问题便能看得更清楚了：零标准差使类别效用计算结果为无穷大。为每个属性强制预设一个最小方差，即灵敏度，是一个简单的问题解决方法。

6.6.4 基于概率的聚类

上述启发式聚类的某些缺点已经很明显：类别效用公式中为避免过度拟合所须采用的k个任意分区，需要提供一个人为的聚类标准差最小值，以及为避免每个实例成为一个聚类的特定的截止值。另外，递增算法本身所带来的不确定性，即结果依赖于实例顺序达到何种程度？诸如合并、分裂这样的局部重建操作，是否足以扭转由不好的实例次序所带来的糟糕的初始决定？最终结果是否代表类别效用的局部最大值？无法知道最终的设定离全局最大值到底有多远，重复几次聚类过程然后选择最好的标准技巧推翻了这个算法的递增本性。最后，结果亦不能回避哪个是最好的聚类这个实质性的问题。图6-18中的聚类有那么多，就像要从糠中筛出小麦那么困难。

解决聚类问题的一个更为理论性的统计学方法可以克服部分上述缺点。从概率的角度看，聚类的目标是寻找给定数据的最有可能的聚类（不可避免地要用到先验期望值）。由于任何有限数量的证据都不足以对某件事做完全肯定的结论，因此，实例甚至是训练实例也不能绝对地被分在这个聚类或那个聚类，而应当说实例都以一定的可能性分属于每个聚类。这有助于消除与那些武断而快速的判定方案相关联的脆弱性。

统计聚类的基础是建立在一个称为有限混合（finite mixtures）的统计模型上的。混合是指用k个概率分布代表k个聚类，掌控着聚类成员的属性值。换句话说，对某个具体实例，每个分布会给出假设它实属这个聚类，实例具有某种系列属性值的概率。每个聚类都有不同的分布。任何具体实例“实际上”属于且只属于一个聚类，但不知是哪个。最后一点，各个聚类并不是同等可能的，存在某种反映它们相对总体数量的概率分布。 262

最简单的有限混合情况是只有一个数值属性，每个聚类是呈高斯或正态分布，但有不同的平均值和方差。聚类问题是获得一系列的实例，这时每个实例只是一个数字和一个事先设定的聚类数目，计算出每个聚类的平均值和方差，以及聚类之间的总体分布。混合模型将几个正态分布组合起来，它的概率密度函数看起来像一组山脉，每座山峰代表一个组成部分。

图6-19展示了一个简单的例子。图中有两个聚类A和B，每个都呈正态分布，聚类A的平均值和方差是μ_A和σ_A；聚类B的平均值和方差是μ_B和σ_B。从这些分布中提取样本，A的提取概率为P_A，B的提取概率为P_B ($P_A+P_B=1$)，得到如图所列的数据集。现在，想象所给的数据集没有类值，只有数据，要求确定模型的5个参数，μ_A、σ_A、μ_B、σ_B和p_A（参数p_B可以直接从p_A计算得到）。这就是有限混合问题。

如果知道每个实例是由哪种分布而来，便很容易找到5个参数值，只要使用下面的公式，分别对A和B 两个样本估计平均值及标准差

$$\mu = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

$$\sigma^2 = \frac{(x_1-\mu)^2 + (x_2-\mu)^2 + \cdots + (x_n-\mu)^2}{n-1}$$

（第二个公式分母用$n-1$而不用n是一种取样技术：在实践中如用n几乎没有什么差别。）这里$x_1, x_2, \cdots, x_n$是取自分布A或B的样本。欲估计第5个参数p_A，只要计算聚类A所含实例数占实例总数的比率。

如果5个参数已知，要找出某个给定实例来自每种分布的概率就很简单了。给定实例x，

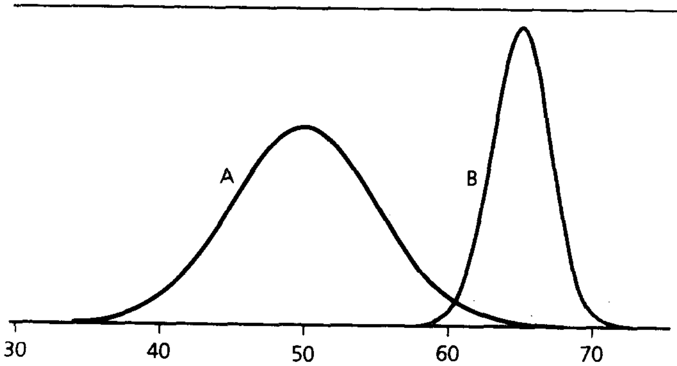
263 它属于聚类A的概率是

数据

A 51	B 62	B 64	A 48	A 39	A 51
A 43	A 47	A 51	B 64	B 62	A 48
B 62	A 52	A 52	A 51	B 64	B 64
B 64	B 64	B 62	B 63	A 52	A 42
A 45	A 51	A 49	A 43	B 63	A 48
A 42	B 65	A 48	B 65	B 64	A 41
A 46	A 48	B 62	B 66	A 48	
A 45	A 49	A 43	B 65	B 64	
A 45	A 46	A 40	A 46	A 48	

a)

模型



$\mu_A = 50, \sigma_A = 5, p_A = 0.6$　　$\mu_B = 65, \sigma_B = 2, p_B = 0.4$

b)

图6-19　一个二类问题的混合模型

$$\Pr[A \mid x] = \frac{\Pr[x \mid A] \cdot \Pr[A]}{\Pr[x]} = \frac{f(x; \mu_A, \sigma_A) p_A}{\Pr[x]}$$

这里，$f(x; \mu_A, \sigma_A)$是聚类A的正态分布函数，即：

$$f(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

分母Pr[x]将消失：计算分子Pr[A|x]和Pr[B|x]，然后要除以两者之和进行正常化。整个过程正如4.2节中朴素贝叶斯学习方案对数值属性所用的处理方法。那里讨论中所作的说明在这里也

同样适用。严格地说，$f(x;\ \mu_A,\ \sigma_A)$并非是概率Pr[x|A]，x等于任何具体数值的概率为零，然而正常化过程使得最终的结果是正确的。注意最终结果不是某个具体的聚类，而是x属于A和B的概率。 264

6.6.5 EM 算法

问题是既不知道每个训练实例来自哪个分布，也不知道混合模型的5个参数值。因此，我们采纳k均值聚类算法的程序，进行迭代。从对5个参数值进行初始猜测开始，用初始猜测值对每个实例进行聚类概率计算，用这些概率对参数进行重新估计，然后重复此过程。（如果愿意，也可以从对每个实例的类进行初始猜测开始。）这种方法称为期望-最大化（expectation-maximization），简称EM 算法。第一步，计算聚类概率（即“期望的”类值）便是“期望”；第二步，计算分布参数，即对给定数据的分布进行似然“最大化”处理。

考虑到已知的只是每个实例所属聚类概率而非聚类本身，因此必须对参数估计公式稍许做一点调整。这些概率作用就像是权值。如w_i是实例i属于聚类A的概率，聚类A的平均值和标准差是

$$\mu_A = \frac{w_1 x_1 + w_2 x_2 + \cdots + w_n x_n}{w_1 + w_2 + \cdots + w_n}$$

$$\sigma_A^2 = \frac{w_1(x_1-\mu)^2 + w_2(x_2-\mu)^2 + \cdots + w_n(x_n-\mu)^2}{w_1 + w_2 + \cdots + w_n}$$

这里x_i不单是属于聚类A的实例，而是包括所有的实例。（这和6.6节所列的估算标准差公式有些许不同。从专业技术上来说，这是一个方差的“最大似然”估计器，而6.6节所列的是一个“无偏”估计器。它们之间的差别在实践中并不重要。）

现在来考虑怎样终止迭代。k均值算法是当实例的类值在下一轮循环中没有变化时即终止，即达到一个“固定点”。在EM算法中，情况并非如此简单，算法会向某个固定点收敛但是却不能真正达到这个点。然而可以通过给定5个参数值来计算数据集数据的总体似然，看出它的靠近程度。总体似然是将单个实例i的概率相乘得到的： 265

$$\prod_i (p_A \Pr[x_i \mid A] + p_B \Pr[x_i \mid B])$$

这里聚类A和聚类B的概率是由正态分布函数$f(x;\mu,\sigma)$决定的。这个总体似然是对聚类“良好性”的一种衡量，在EM的每次迭代中不断上升。这里又出现了x为某个具体数值的概率等于$f(x;\mu,\sigma)$的技术难题，由于没有对概率进行正常化操作，这个影响并没有消失。结果是上面的似然表达式代表的不是概率，不必一定要在0和1之间。然而，它的大小仍然反映了聚类质量的好坏。在实现中一般取它的对数：只要计算每个组成部分的对数总和，避免了相乘的计算。总体结论还是保持不变，循环迭代直到对数似然（log-likelihood）的增值可忽略不计。例如，在一个实际的实现中，可以循环迭代直至出现连续10次迭代前后两轮对数似然的差值小于10^{-10}。一般来说，前面几轮迭代的对数似然会急剧上升，然后快速收敛于某个几乎是固定的点。

虽然EM算法能保证收敛于某个最大值，但也许只是局部最大值而非全局最大值。为了能有机会得到全局最大值，整个过程必须使用不同的初始猜测参数值重复几次。可以用总体对数似然数值来比较不同的参数配置，只要选择其中最大的。

6.6.6 扩展混合模型

已经看了含两个高斯分布的混合模型，现在来考虑一下怎样将其扩展到更现实的情况中。基本方法是相同的，但由于数学表达更令人恐怖，在这里就不全面展开了。

只要正态分布数量k事先已知，将适用于二类问题的算法转变为适合解决多类问题是非常直截了当的。

只要存在属性独立假设，适合于单个属性实例的模型可以扩展为适合于多个属性实例的
266 模型。就像朴素贝叶斯方法那样，将实例每个属性的概率相乘得到这个实例的联合概率。

当已知数据集含有关联属性，独立假设便不再成立。两个属性可用二维正态分布建立联合模型，每个分布有各自的平均值，但采用含4个数值参数的"协方差阵"（covariance matrix）来替换两个标准差。有标准的统计技术用以估计实例的类概率，已知实例和类概率可以估计出平均值和协方差阵。对多个关联属性的，可以使用多维分布来处理。参数的数量随着联合属性数量的平方而增加。对于n个独立属性，有$2n$个参数，各含一个平均值和一个标准差。对于n个共变属性（covariant attributes），有$n+n(n+1)/2$个参数，各含一个平均值和一个$n\times n$协方差阵，这个矩阵是对称的，因此有$n(n+1)/2$个不同的数值。像这样的参数数量增长将造成严重的过度拟合，我们将在下面讨论。

为了适应名词性属性，必须放弃正态分布。对一个含有v个可能值的名词性属性，用v个数字来代表每种值的概率。对每个类需要有不同的数字组合；总共有kv个参数。这个情形同朴素贝叶斯方法很相似。对应的期望和最大化这两个步骤同先前所述的操作是一样的。期望：给定分布参数，对每个实例所在的聚类进行估计，如同对未知实例进行类预测。最大化即用已分类的实例对参数进行估计，如同从训练实例中决定属性值概率，一个小区别在于EM算法实例被赋予的是类概率而非类别。在4.2节中已遇到估计概率可能为零的问题，这里也同样会碰到。所幸的是解决方法很简单，使用拉普拉斯（Laplace）估计器。

朴素贝叶斯有独立属性的假设，这是它称为"朴素"的原因所在。一对关联名词性属性，可能的属性值分别有v_1个和v_2个，可以用单个共变属性来替代，它的可能属性值有v_1v_2个。参数的数量随着关联属性的数量增加，这将牵连到概率估计及随后即将讨论的过度拟合问题。

对既有数值属性又有名词性属性的数据进行聚类没有什么特别的问题，共变的数字和名词性属性处理起来更加困难，这里不作讨论。

残缺值可以使用多种不同方法来调整。名词性属性的残缺值可以如4.2节中所述的那样不
267 参与概率计算；或者也可以把它们当作是另一个属性值，和其他属性值一样对待。哪种方法比较合适取决于"残缺"的含义。对于数值属性，则采用相同的概率。

有了这些改进，概率聚类变得相当完备。EM算法贯穿于整个工作过程中。用户必须确定聚类数目、每个属性的类型（数值属性或名词性属性）、哪些属性要应用共变模式以及如何处理残缺值。另外，除了上述分布类型外，还可应用其他不同的分布。虽然对于数值属性正态分布通常是一个好的选择，但对于某些属性它并不适合，那些属性（譬如权）有预设最小值（对于权来说为零）却没有上限，这时比较适合使用"对数－正态"（log-normal）分布。同时具有上限和下限的数值属性可以用"对数－概率"（log-odds）分布。属性值为整数而非实数时最好使用泊松分布（poisson distribution）。一个完善的系统应能允许对每个属性进行个别设定。在每种情形下，分布都要牵涉到数字参数：对于离散属性来说是所有可能属性值的概率，对于连续属性来说是平均值和标准差。

在这节中讨论的是聚类。也许你会想到这些改进措施也应能很好运用于朴素贝叶斯算法，你是对的。一个完善的概率模型适用于聚类学习和分类学习，适用于各种不同分布的名词性属性和数值属性，适合于各种不同的共变可能，也适合于不同的残缺值处理方法。作为领域知识的一部分，用户要确定各个属性所使用的分布。

6.6.7 贝叶斯聚类

然而，还有一个障碍：过度拟合。你也许会说如果不确定哪些属性是互相依赖的，为什么不安全一点将所有属性都设定为共变的？答案是参数越多最终模型结构越是可能对训练数据产生过度拟合，共变设定使参数数量急剧上升。机器学习中总是产生过度拟合问题，概率聚类也不例外。产生的途径有两条：所设聚类的数目太多，以及所设分布参数太多。

一种太多聚类数目的极端现象是每个数据点即为一个聚类：很清楚这将产生对训练数据的过度拟合。实际上，在混合模型中，每当正态分布变得很狭窄以至于集中在一个数据点上， 268
就会产生过度拟合问题。因此在实现中，通常要规定聚类至少必须含两个不同数据值。

只要参数数量多，就会发生过度拟合问题。如果不确定哪些属性是共变的，你可能会对各种不同共变的可能进行试验，然后挑选其中能使数据处于所找到聚类的总体概率达到最大值的那个。不幸的是，参数数量越多，这个数据总体概率也越高。这个高概率并非是好的聚类所需的，而是过度拟合所需的。越多的参数参与进来，越是容易找到看起来似乎很好的聚类。

如果能在引入新参数的同时，对模型设置某些障碍是个不错的主意。一个基本的方法就是完全采纳贝叶斯方法，让每个参数都有一个先验概率分布。然后每当引入一个新参数，它的先验概率要参与总体似然的计算。由于总体似然要乘以一个小于1的数字，即先验概率，这便给因新参数引入而产生的似然增值自动设障。若要提高总体似然，新参数必须能产生超出惩罚的收益。

从某种意义上看，4.2节中提到的以及上述对于名词性属性值遇到零概率问题时，建议使用的拉普拉斯估计器是这样一种机构，每当观察到的概率很小时，拉普拉斯估计器强制设障，使这个零或接近零的概率提高，从而降低数据的总体似然。将两个名词性属性共变会加剧这个问题。原先有v_1+v_2个参数，这里v_1和v_2是可能的属性值个数，现在增为v_1v_2个参数，同时也大大增加了产生大量小估计概率的机会。实际上，拉普拉斯估计器与引入新参数时使用某个特定的先验概率是等效的。

对于聚类数目过多的问题也可采用相同的技术来抑制，只要预设一个先验分布，当聚类数目增加时它将急剧下降。

AutoClass是一种完善的贝叶斯聚类方法，它使用有限混合模型，每个参数都带有先验分布。它适用于数值属性和名词性属性，并且使用EM算法对概率分布参数做出最适合数据的估计。由于不能保证EM算法一定收敛于全局最佳点，因此使用不同的初始值进行多次重复运行。不仅如此、AutoClass还考虑到不同的聚类数目，不同的协方差，以及对于数值属性的不同概率分 269
布类型。这又牵涉到一个额外的外层搜寻。例如，它初始时分别对2、3、5、7、10、15和25个聚类进行对数-似然评估，然后为结果数据找到合适的对数-正态分布，并从中随机抽取，用更多的值进行测试。正如你所想像的，整个算法非常耗时。在实际实现过程中，有一个预设的时间限度，只要在时间允许范围，便继续迭代过程。这个时间限度设得越长，效果越好。

6.6.8 讨论

上述各种聚类方法产生不同的输出。它们都能对新数据以测试集的形式，根据对训练集

分析所得的聚类来进行分类。然而，只有递增聚类方法可以生成清晰的知识结构，能够将聚类描述可视化并做出合理的论述。至于其他算法形成的聚类，如果维数不是太多的话，可在实例空间实现可视化。

如果使用聚类方法对训练集实例按照所在聚类编号赋予标签，标有标签的数据集便可用于规则训练或决策树学习器。规则或决策树的学习结果将形成清楚的类描述。概率聚类方案也可这样应用，除了每个实例可能有多个加权标签，因此规则或决策树学习器必须能处理加权的实例，许多都可以。

聚类的另一个应用是填补属性的残缺值。例如，可以对某个实例的未知属性值进行统计估计，根据实例本身的类分布以及其他样本中这个未知属性值来估计。

我们所考查的所有聚类方法都是在独立属性这个假设前提下的。AutoClass允许用户事先设定两个或两个以上的属性间存在相互依赖关系，并用联合概率分布模型。（然而，这里有个限定：名词性属性间也许会发生关联变化，数值属性间同样也会，但这两种属性之间不可关联变化。另外，对于残缺值，关联变化属性也不适合。）使用某种统计学技术，譬如将在7.3节中讨论的主分量转换法（principal components transform），对数据集进行预处理从而使属性更加独立，或许能有些益处。注意，使用这种技术，并不能消除存在于某些特定类内的联合变化，它只能消除存在于所有类之间的总体联合变化。

通过重复分裂聚类并考察分裂是否值得，以此来改进k均值法使之能找到较好的k值，这个
270 方法是根据Moore和Pelleg（2000年）的X均值算法。它使用了一个称为贝叶斯信息标准（Bayes Information Criterion）的概率方案（Kass和Wasserman，1995年）来替代最短描述长度原理。基于合并和分裂操作的递增聚类程序是在适用于名词性属性的Cobweb系统（Fisher，1987年）和适用于数值属性的Classit系统（Gennari等，1990年）中提出的。这两个系统都是以1985年Gluck和Corter定义的类别效用衡量方法为基础的。AutoClass程序是在1995年由Cheeseman 和Stutz 提出的。现有两种实现版本：用LISP 实现的原始研究版本和随后用C编写的版本，后者要比前者快10到20倍，但却有些局限，例如对于数值属性只实现了正态分布模型。

6.7 贝叶斯网络

4.2节中的朴素贝叶斯分类器以及4.6节中的logistic回归模型都是产生概率估计来替代类预测的。对于每个类值，它们都是估计某个实例属于这个类的概率。大多数其他类型的分类器如有必要都可以强制产生这类信息。例如，通过计算叶节点上每个类的相对频率，便能从决策树中得到概率，同样地通过检验某条规则所覆盖的实例，便能从决策列中得到概率。

概率估计经常比仅仅是预测更为有用。它们可以对所做的预测进行排名，使期望成本达到最小化（见5.7节）。事实上，把分类学习当作是从数据中学习类概率估计的任务来完成，还存在很大的争议。所估计的是类属性值的条件概率分布，前提是给定其他属性值。分类模型是用一种简洁易懂的形式来表达这个条件分布的。

由此看来，朴素贝叶斯分类器、logistic回归模型、决策树等等只是用不同的方法表达一个条件概率分布。当然，它们的表达能力有所差别。朴素贝叶斯分类器和logistic回归模型只能表达较简单的分布，而决策树却能表达至少可以近似表达任意分布（arbitrary distribution）。但决策树也有缺陷：它们将训练集分隔成越来越小的数据集，必然造成概率估计可靠性的下降，并且还存在3.2节中提到的重复子树问题。规则集似乎可以克服这些缺点，但是引导一个

好的规则学习器设计所采用的启发式方法尚缺乏理论依据。 271

这是否意味着只能认命，让这些缺陷继续存在？不！有一种基于统计理论的方法：具有较强理论根基、采用图解方式简洁易懂地表达概率分布的方法。这个结构称为贝叶斯网络(Bayesian networks)。画出的图形就像是节点网络图，每个节点代表一个属性，节点间用有向连线连接着，却不能形成环，一个有向无环图（directed acyclic graph)。

在解释贝叶斯网络是如何工作的以及怎样从数据中来学习贝叶斯网络时，要做一些简化假设。假设所有属性都是名词性的并且没有残缺值。有些高级的学习算法可以产生包含在数据之外的新属性称为隐蔽属性，它们的属性值是看不到的。如果这些属性表现了潜在问题的特征，便能支持产生更好的模型，而贝叶斯网络提供了一个很好的方法能在预测时使用这些属性。然而，这使得学习和预测更为复杂并且耗时，因此这里不作讨论。

6.7.1 做出预测

图6-20展示了一个关于天气数据的贝叶斯网络的简单实例。图中数据的4个属性阴晴、温度、湿度和刮风以及类属性玩分别用节点表示。有向连线从玩节点指向其他各个节点。但在贝叶斯网络中图形结构只是一部分，在图6-20中每个节点内还含有一个列表。表中的信息定义了将用于预测任意一个实例类概率的概率分布。

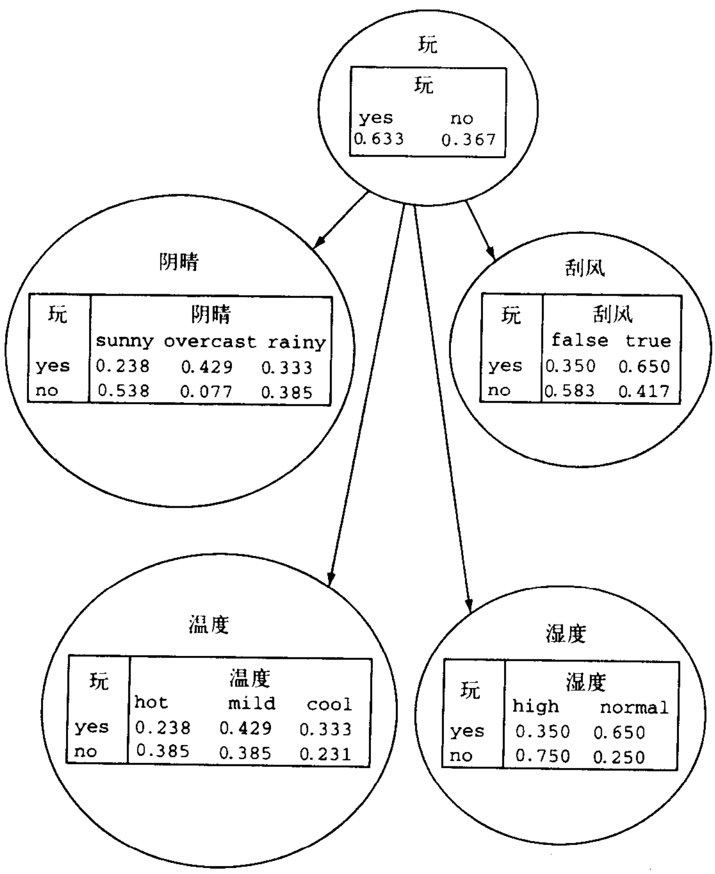


图6-20 天气数据的一个简单贝叶斯网络实例

在讨论怎样计算这个概率分布之前，先来考虑一下表中的信息。下方的4个表（阴晴、温度、湿度和刮风）被一根垂直线划分成两个部分。左边是玩属性的属性值，右边对应的是这个节点所代表属性的各个属性值的概率。一般来说，左边包含的每列分别代表每个指向该节点的属性，在这里只有玩属性。这也是玩节点本身列表中左边没有信息的原因：它没有父辈节点。一般来说，每一行的概率对应于一组父辈属性值组合，行中的每个概率代表了这个节点属性的每个属性值对应于某种属性值组合的概率。实际上，每一行都是对该节点属性的属性值的一个概率分布定义。每行中各个概率的总和始终是1。

图6-21展示了同是这个问题的一个更为复杂的网络，这时其中3个节点（刮风、温度和湿度）都有两个父辈节点。同样，每个父辈都会在左边产生一列，右边的列数等同于节点属性的属性值数量。考虑温度节点所含列表的第一行。左边列出了它的每个父辈属性的属性值，玩和阴晴；右边列出了每个温度属性值的概率。例如，第一个数字（0.143）是当玩和阴晴的属性值分别为yes和sunny时，温度的属性值为hot的概率。

272
~
273
怎样利用这些表来预测某个实例的每个类值的概率呢？由于假设没有残缺值，这就变得很简单了。实例的每个属性都有确定的属性值。对网络中的每个节点，根据父辈属性值找到相应的行，查看该行节点属性值的概率。然后将这些概率相乘。

例如，考虑这样一个实例阴晴 = rainy, 温度 = cool, 湿度 = high, 刮风 = true 。要计算玩 = no 的概率，查看图6-21中的网络，玩节点给出概率0.367，阴晴节点给出概率0.385，温度节点给出概率0.429，湿度节点给出概率0.250，刮风节点给出概率0.167。它们的乘积是0.0025。对玩 = yes 也做相同的计算，得到0.0077。很显然这并非是最终答案：最终概率之和应为1，而现在0.0025和0.0077之和不等于1。实际上它们是Pr[玩 = *no*, *E*] 和Pr[玩 = *yes*, *E*]的联合概率，其中*E*是指由这个实例的属性值所给出的所有例证。联合概率既衡量一个实例各自类值，也衡量实例的属性值在*E*中的似然。只有考虑了包括类属性在内的所有可能的属性值组合空间时，它们的和才为1。这个例子绝对不属于这种情况。

解决方法很简单（在4.2节中曾遇到过）。要得到条件概率Pr[玩 = *no* | *E*] 和Pr[玩 = *yes* | *E*]，只要将这两个联合概率正常化处理，即分别除以两者之和。得到玩 = no的概率为0.245，玩 = yes的概率为0.755。

只剩下一个迷了：为什么是将这些概率相乘呢？乘积步骤的有效性是要有一个前提假设的，那便是给定每个父辈节点所代表属性的属性值，知道任何其他先辈的属性值并不能使该节点属性的各个可能的属性值所对应的概率发生变化。换句话说，就是先辈并不能提供任何多于父辈所能提供的有关该节点属性值的似然信息。这点可以表示为

$$\Pr[\text{节点}|\text{祖先}]=\Pr[\text{节点}|\text{父节点}]$$

所有节点值和牵涉在内的属性都必须遵守这条假设。在统计学中，这称为条件独立(conditional independence)。给定父辈属性，每个节点对于它的祖父辈、曾祖父辈等等都是条件独立的，在这种情形下乘积便是有效的。乘积步骤遵循概率理论中的链规则，链规则提出n个属性a_i的联合概率可以分解为如下的乘积：

$$\Pr[a_1, a_2, \cdots, a_n] = \prod_{i=1}^{n} \Pr[a_i \mid a_{i-1}, \cdots, a_1]$$

这个分解表达式对于任何一种属性排列都是成立的。因为贝叶斯网络是一种无环图，可以将

网络节点进行排列，使节点a_i的所有先辈序号都小于i。然后，由于有条件独立的假设，

$$\Pr[a_1, a_2, \cdots, a_n] = \prod_{i=1}^{n} \Pr[a_i \mid a_{i-1}, \cdots, a_1] = \prod_{i=1}^{n} \Pr[a_i \mid a_i \text{的父节点}]$$

正是前面所应用的乘积规则。

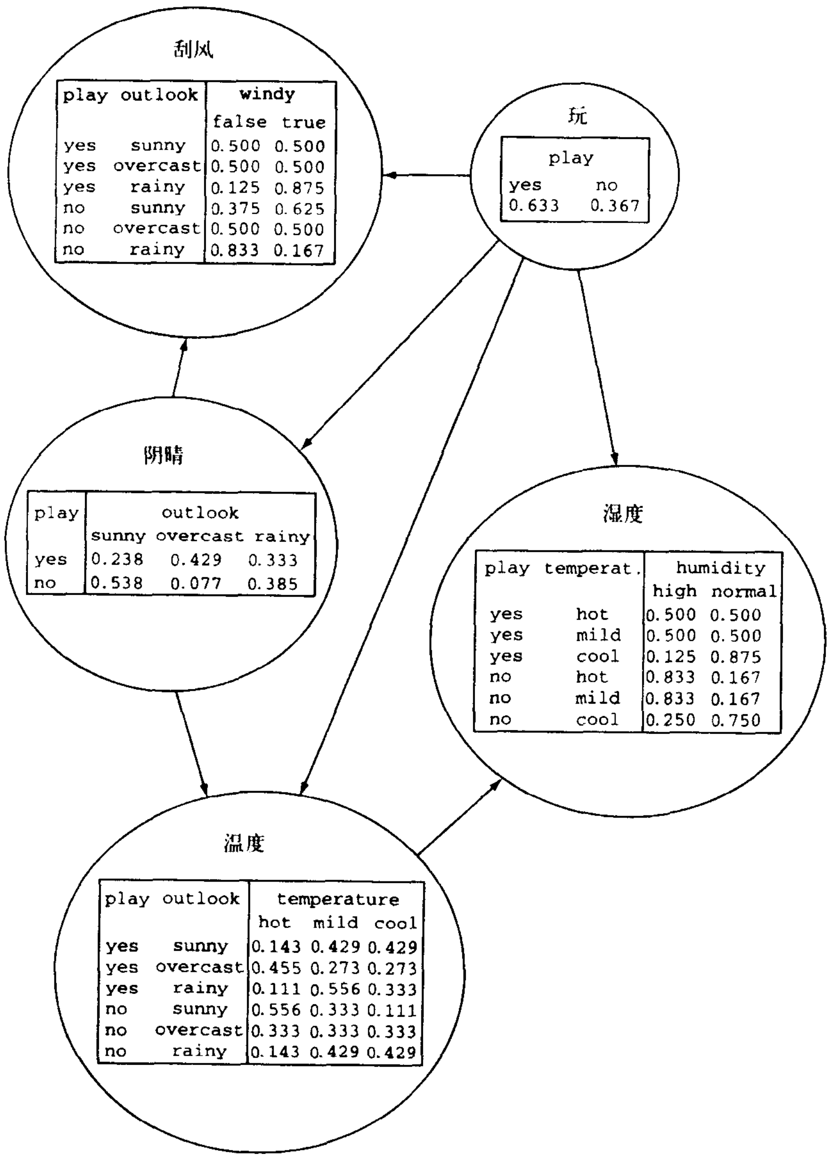


图6-21 天气数据的另一个贝叶斯网络 274~275

图6-20和图6-21展示的两种贝叶斯网络从本质上来看是不同的。第一个（图6-20）具有更为严格的独立假设，因为它的每个节点的父辈都是第二个（图6-21）中所对应的每个节点的父辈的一个子集。实际上，图6-20几乎就是4.2节中的朴素贝叶斯分类器（概率计算略有不同，只是因为每个累计数都被初始为0.5以避免零频率问题）。图6-21网络中的条件概率列表含有更

多的行，因此需要更多的参数；它可能会是对于潜在领域更为精确的一种表示法。

假设贝叶斯网络中的有向连线组代表的是因果关系，这是个非常诱人的假设。要谨慎！在上述例子中，玩属性的属性值会使阴晴属性的某个具体值所对应的期望值提高，但事实上两者并没有因果关系，也许更有可能是相反的。对具有同样概率分布的相同问题，可以建立不同的贝叶斯网络。可以采用不同方法，利用条件独立对联合概率分布进行分解来实现。使用有向连线反映因果关系模式的网络是最为简单的，它所带的参数最少。因此为某个特定领域组建贝叶斯网络的专家们经常受益于用有向连线来代表因果关系。然而，机器学习技术是从数据中导出模型，其中的因果结构是未知的，所能做的只是根据从数据中观察到的相关性建立网络。而从数据相关性推导出的因果关联总是风险很大的。

6.7.2 学习贝叶斯网络

为贝叶斯网络建立一种学习算法的方法是定义两个组成部分：一个是对基于某个数据集的网络进行评估的评估函数，另一个是在所有可能的网络空间中搜索的搜索方法。某个给定网络的质量是根据对于给定网络的数据的概率来衡量的。我们计算网络和每个实例相符和的概率，然后将所有实例的这些概率相乘。在实践中，很快会使这个数字达到非常小，以至于不能较好地反映质量（称为算法下溢 arithmetic underflow），因此，我们使用概率对数的总和来替代原先的乘积。最终的计算结果就成了对于给定数据，网络的对数-似然。

假设网络结构即有向连线组是已知的。很容易估计条件概率表中所列的数字，只要计算训练数据中对应属性值组合的相对频率。为避免零频率的出现，像4.2节所述的那样使用一个常量来初始化累计数。例如，为要找出已知玩 = yes 和温度 = cool 时，湿度 = normal 的概率
276 （图6-21的湿度节点列表中第三行的最后一个数字）。从表1-2中，可观察到天气数据中有3个实例含有这样的属性值组合，却没有实例湿度 = high，同时含相同的玩和温度属性值。将湿度两个属性值的累计数初始为0.5，得到湿度 = normal的概率为（3+0.5）/(3+0+1) = 0.875。

网络中的节点是预设的：每个属性各有一个（包括类属性）。学习网络结构等于是在可能的有向连线组空间进行搜索，对每组的条件概率表进行估计，并计算结果网络基于某个数据集的对数-似然，以此作为对网络质量的衡量。各种贝叶斯网络学习算法的不同之处在于它们在网络结构空间的搜索方式。以下是一些算法的介绍。

有一点要告诫的。如果对数-似然是基于训练数据集进行了最大化，增加更多的有向连线总是会获得较好的结果，造成最终的网络过度拟合。很多方法可用于解决这个问题。一种可能的方法是采用交叉验证法来估计拟合的良好度。第二种是根据参数数目，相应增加网络复杂度惩罚，即在所有概率列表中独立估计的总数目。每个表中，独立概率的数目是表中所有概率的总个数减去最后一列中的个数，由于每一行概率的和应为1，因此最后一列的概率可根据其他列的值而导出。假设K为参数数量，LL表示对数-似然，N为数据集中的实例数量，有两个较普及的衡量方法可用于评估网络质量：Akaike 信息标准（Akaike Information Criterion，简称AIC）

$$\text{AIC score} = -LL + K$$

以及下面所列基于最短描述长度原理的MDL度量：

$$\text{MDL score} = -LL + \frac{K}{2}\log N$$

这两个表达式中对数-似然都是负数，因此目标便是使它们的结果得分最小化。

第三个可能的方法是赋予网络结构一个先验分布，通过组合先验概率以及数据与网络相符的概率，找出可能性最大的网络。这是网络评分的“贝叶斯”法。取决于所使用的先验分布，可以有多种形式。但是，真正的贝叶斯要平衡所有可能的网络结构，而不是只取其中某 277
个具体网络进行预测。不幸的是这需要进行大量的计算。一个简化了的方案是平均某个给定网络的所有子结构网络。改变计算条件概率表所采用的方法，从而使结果概率估计包含了来自所有子网络的信息，这个方案实现起来非常有效。这个方案的细节也相当复杂，这里就不再讨论了。

如果使用适当的评分度量，搜寻一个好的网络结构的任务可以大大地简化。回顾一下一个网络中某个实例的概率是所有条件概率表中单个概率的乘积。数据集的总体概率是把所有实例的这些乘积再进行乘积运算得到的。由于乘积运算中的项是可互换的，因此这个乘积表达可以重写为把同属一个表的各个系数组合在一起的形式。在对数-似然的计算中，是将求和运算代替乘积运算，因此也同样可以如此处理。这意味着似然的优化可以在网络的每个节点中分别进行。它可以通过增加或去除其他节点指向正在进行优化的节点的连线来完成，唯一的限制就是不可引入环。如果局部评分度量使用如AIC或MDL的方案来替代朴素的对数-似然，这个诀窍也同样生效，因为惩罚项会被分裂成几个组成部分，每个节点各一个，而且每个节点可以进行独立的优化。

6.7.3 算法细节

现在来看用于贝叶斯网络学习的实际算法。一个简单而快速的学习算法称为*K2*起始于某个给定的属性（即节点）排序。然后，对每个节点依次进行处理，贪心地增加从先前处理过的节点指向当前节点的连线。每一步过程中都增加那些能使网络评分达到最高值的连线。当不再有改进时，便将注意力转向下一个节点。每个节点的父辈数量可以被限制在一个预设的最大值范围内，这可作为一个防止过度拟合的附加机构，由于只考虑起始于前面已经处理过的节点的连线，并且顺序是固定的，这个程序不会产生环。但是结果依赖于初始排序，因此采用不同的随机排序对算法进行多次运作是有意义的。

朴素贝叶斯分类器是这样一种网络，它的连线是由类属性指向其他每个属性。当为分类而建立网络时，使用这种网络作为搜寻起点有时还是有帮助的。可以使用*K2*方案来实现，强制把类属性列为序列中的第一个属性并合理地设定初始连线。

另一个潜在的有用技巧就是要确保数据的每个属性都在类属性节点的马尔可夫毯（Markov blanket）范围内。一个节点的马尔可夫毯包含该节点的父辈节点、子节点以及子节点的父辈节点。可以证明节点与它的马尔可夫毯内的所有其他节点都是条件独立的。因此，如果一个节点不包括在类属性的马尔可夫毯内，那么该节点所代表的属性与分类就是毫无关 278
系的。反过来，如果*K2*发现一个网络没有将某个相关属性包含在类属性的马尔可夫毯内，或许可以增加一条连线来纠正这个缺点。一个简单的方法便是增加一条连线，自这个属性节点指向类节点，或自类节点指向这个属性节点，这要取决于哪种方向可以避免环。

不进行节点排序，而是采取贪心考虑增加或去除任意一对节点间的连线（当然始终要保证无环）是一个更为周全但却较慢的K2版本。再进一步就是要同时考虑反转现存连线。使用任何的贪心算法，结果网络只能代表某个局部最大评分函数，通常建议采用不同的随机初始

值，多次运行算法。还可使用如模拟退火（simulated annealing）、tabu 搜索或是遗传算法（genetic algorithms）等更为复杂的优化策略。

另一种较好的贝叶斯网络分类器学习算法称为树扩展型朴素贝叶斯（tree augmented Naïve Bayes（TAN））。正如它的名称所暗示的，在朴素贝叶斯分类器上添加连线。类属性是朴素贝叶斯网络每个节点的单一父辈节点，TAN考虑为每个节点增加第二个父辈节点。如果排除类节点和其相应的所有连线，假设只有一个节点没有增加第二个父辈节点，结果分类器包含一个以无-父辈（parent less）节点为根节点的树结构——这也是这个节点名称的由来。对于这类限制型的网络有一种有效的算法能找出使网络似然达到最大值的连线组合，它是以计算网络的最大加权生成树为基础的。这个算法与实例数量呈线性关系，与属性数量呈二次函数关系。

到目前为止所讨论的评分度量都是基于似然的，意义在于使每个实例的联合概率$\Pr[a_1, a_2, \cdots, a_n]$最大化。然而，在分类问题中，真正要最大化的是在给定其他属性值的情况下，类的条件概率，换句话说就是条件似然。不幸的是，对于贝叶斯网络的列表中所需要的最大条件似然概率估计没有闭合形式的解决方法。另一方面，为某个给定的网络和数据集，计算条件似然是很直截了当的，最终是logistic 回归所要做的。因此有人提议在网络中使用标准的最大似然概率估计，而对某个具体的网络结构则采用条件似然来评估。

另一种使用贝叶斯网络进行分类的方法是为每种类根据属于该类值的数据分别建立网络，
279 并利用贝叶斯规则来组合这些网络预测结果。这一组网络称为贝叶斯复网（Bayesian multinet）。要得到某个类值的预测，将相应网络的概率乘以类值的先验概率。对每个类都进行如此操作并将结果进行正常化处理。在这种情形下，不采用条件似然来进行每个类值的网络学习。

上述所有网络学习算法都是基于评分的。另一种策略，是通过对各个基于属性子集的条件独立性进行测试，拼凑起一个网络来，这里不作讨论。这就是所谓的“条件独立测试的结构学习”（structure learning by conditional independence tests）。

6.7.4 用于快速学习的数据结构

学习贝叶斯网络涉及到大量的计算。对于搜索过程中考虑的每个网络结构，数据必须被重新扫描，以获得填写表中的条件概率所需的累计数。是否可以将它们保存为某种数据结构以消除一次又一次重新扫描数据的需要呢？一种显而易见的方法就是预先计算好累计数并将非零数值保存在一个表中，比如4.5节中提到的散列表。即使如此，任何非平凡的数据集都将会产生大量的非零累计数。

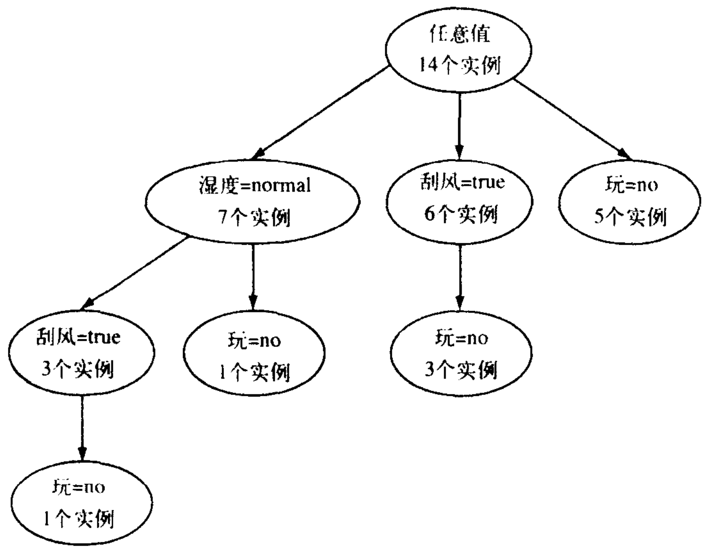
再次考虑表1-2的天气数据。表中列有5个属性，其中2个属性各含3个属性值，另外3个属性各含2个属性值。这便给出$4 \times 4 \times 3 \times 3 \times 3 = 432$个可能的累计数。乘积中的每个部分各对应于一个属性，各自的贡献要比它所含属性值数量多一个，这是因为在累计中这个属性可以没有。所有这些累计计算可以当作如4.5节中所描述的项集来处理，最小覆盖数量设为1。即使不保存零累计数，运行这个简单的方案也会很快带来内存问题。

可以用一种称为AD（全维）树（all-dimensions tree）的结构来有效地储存累计数，它类似于4.7节所描述的用于最近邻搜索的kD树。为了简单起见，这里使用简化版的天气数据来进行描述，简化数据只含有湿度，刮风和玩这三个属性。图6-22a列举了数据。虽然图中只列出了8组数据，但可能的累计数数量有$3 \times 3 \times 3 = 27$个。例如，玩 = no 的累计数是5（累加起来！）。

图6-22b展示的是这个数据的AD树。每个节点显示了从树根到该节点沿途共有多少实例，它们的属性值经过了测试。例如，最左端的叶节点显示有1个实例，它的湿度 = normal，刮风 = true 以及玩 = no，而最右端的叶节点显示有5个玩 = no的实例。 280

湿度	刮风	玩	累计
high	true	yes	1
high	true	no	2
high	false	yes	2
high	false	no	2
normal	true	yes	2
normal	true	no	1
normal	false	yes	4
normal	false	no	0

a）简化版数据



b）对应的AD树

图6-22 天气数据

将所有27种累计情形都清楚地罗列出来的树没有什么意义。然而，它只包含了8个累计数，它所能获得的不比表多，很显然这并非是图6-22b所要表达的树。例如，没有能测试湿度 = high 的分支。树是怎样建起来的，怎样才能从中得到所有的累计数呢？

假设数据的每个属性赋予一个索引。在简化版天气数据中，赋予湿度索引为1，刮风索引为2，玩索引为3。AD树的是这样形成的，每个节点都有一个对应的属性i，该节点是根据索引$j > i$的所有属性的属性值进行扩展的，它有两个重要的限制：对每个属性涉及面最广的扩展省略不做，同样累计数为零的扩展也省略不做。根节点的索引为0，因此对于根节点来说，所有属性都得到扩展，但同样受限于上述两个限制。 281

例如，图6-22b中根节点没有对刮风 = false 进行扩展，因为有8个这样的实例，是涉及面最广的扩展：数据中出现false 值的次数大于出现true值的次数。类似地，节点湿度 = normal 也没有对刮风 = false 进行扩展，因为湿度 = normal的所有实例中刮风为 false值是最为普遍的。

实际上，在这个例子中第二条限制，即累计数为零的扩展省略不做不会用到，因为第一条限制已经排除了任何以湿度 = normal 以及刮风 = false为开始的测试，而这正是图6-22a中唯一累计值为0的情形。

树的每个节点代表某种具体属性值组合的发生。从树中获取某种组合出现的累计数是非常直接的方法。但是由于每个属性最普遍的扩展都被省略，造成许多非零累计数不能在树中清楚地表现出来。例如，湿度 = high和玩 = yes 的组合在数据中出现了3次，但在树中却没有这个节点。然而，结论是任何累计数都可以明确地由储存在树中的累计数计算获得。

这里有个简单的例子。图6-22b中没有包含湿度 = normal，刮风 = true，玩 = yes 的节点。但是，它显示了3个湿度 = normal并且刮风 = true的实例，其中有1个实例玩 属性值不是yes 。这说明必有2个实例玩 = yes 。现在来看一个更为巧妙的情形。湿度 = high, 刮风 = true, 玩 = no 出现多少次？一眼看上去似乎不可能知道答案，因为根本就没有湿度 = high的分支。然而，可以用刮风 = true，玩 = no的累计数(3) 减去湿度 = normal, 刮风 = true，玩 = no的累计数（1）推导出来。从而得出正确累计值是2。

这个方法适用于任何属性分支以及任何属性值组合，但也许需要递归应用。例如，要得到湿度 = high，刮风 = false，玩 = no的累计数 ，需要知道刮风 = false，玩= no的累计数以及湿度 = normal，刮风 = false，玩 = no的累计数。前者可以通过从玩= no的累计数（5）中减去刮风 = true，玩 = no的累计数（3）而获得，结果是2。后者可以通过从湿度= normal，玩 = no的累计数 (1) 中减去湿度= normal, 刮风 = true，玩 = no的累计数（1）而获得，结果为0。因此正确答案一定是有2 − 0 = 2 个实例 湿度 = high，刮风 = false, 玩 = no。

只有当数据含有成千个实例时，才能显示AD 树的价值所在。很明显对于天气数据AD 树并没有帮助。它们不能有益于小的数据集的事实意味着在实践中树结构向所有方向扩展一直
282 到叶节点是没有多大意义的。通常可以应用一个截止参数k，在覆盖实例数量少于k的节点上，保留的是指向这些实例的指针序列，而不是指向下其他节点的指针序列。从而使树结构更小、效率也更高。

6.7.5 讨论

用于贝叶斯网络学习的K2算法是由Cooper 和 Herskovits（1992年）介绍的。贝叶斯评分度量是Heckerman 等在1995年提出的。TAN算法是Friedman 等在1997年提出的，他还对复网进行了讨论。Grossman和Domingos（2004年）展示了怎样利用条件似然为网络进行评分。Guo和Greiner（2004年）对贝叶斯网络分类器的评分度量进行了广泛的比较。Bouckaert (1995年) 对平均子网络进行了探讨。AD树是由Moore和Lee（1998年）介绍并分析的，在4.9节中介绍的kD树和球树也是由Andrew Moore提出的。在近期文献中，Komarek和Moore（2000年）介绍了用于递增学习的AD树，这种树对于含很多属性的数据集也能有更高的效率。

我们只是粗粗掠过贝叶斯网络学习的表层。留下残缺值、数值属性以及隐蔽属性都还没有讨论，也没有讲述怎样在回归任务中使用贝叶斯网络。贝叶斯网络是广泛的统计模型中的特殊一类，称为图形模型，也包括无向连线的网络（称为马尔可夫网络）。近年来机器学习社
283 团对图解模型（graphical model）也赋予了极大关注。

第7章　转换：处理输入和输出

在前一章中我们考察了大量的机器学习方法：决策树、决策规则、线性模型、基于实例的方案、数值预测技术、聚类算法以及贝叶斯网络。所有这些方法都是合理、成熟的技术，可用于解决实际的数据挖掘问题。

但是成功的数据挖掘远不只是牵涉到选择某种学习算法并应用于数据。许多学习算法要用到各种不同的参数，需要选择合适的参数值。在多数情况下，选择适当的参数可以使所获结果得到显著改善，而合适的选择则是要视手头的具体数据而定的。例如，决策树可以选择修剪或不修剪，选择前者又需要选择修剪参数。在基于实例的k最近邻学习方法中，则需要选择k值。更为常见的，则是需要从现有的方案中选择学习方法本身。在所有情况下，合适的选择是由数据而决定的。

在数据上试用几种不同的方法，并使用几种不同的参数值，然后观测哪种情况结果最好，是个诱人的方法。不过要当心！最佳选择并不一定是在训练数据上获得最好结果的那个。我 285
们曾反复提醒要注意过度拟合问题，过度拟合是指一个学习模型与用于建模的某个具体训练数据集太过匹配。假设在训练数据上所表现的正确性能代表模型将来应用于实践中的新数据上的性能水准，这个想法是不正确的。

所幸的是在第5章中已经讨论了对于这个问题的解决方法。有两种较好的方法可用来估计一个学习方法的预期真实性能表现：在数据源充足的情况下，使用一个与训练数据集分离的大数据集；在数据较少的情况下则使用交叉验证法（第5.3节）。在后一种情况下，在实践中的典型应用方法是单次的10折交叉验证，当然要得到更为可靠的估计需要将整个过程重复10次。一旦为学习方法选定了合适的参数，就可以使用整个训练集（即所有训练实例）来生成将要应用于新数据的最终学习模型。

注意在调整过程中使用所选的参数值得到的性能表现并不是对最终模型性能的一个可靠估计，因为最终模型对于调整中使用的数据有过度拟合的倾向。要确定它的性能究竟如何，需要另外一个大的数据集，这个数据集须与学习过程和调整过程中所使用的数据隔离开来。在进行交叉验证时也是如此，参数调整过程需要一个“内部”交叉验证，误差估计还需要一个“外部”交叉验证。采用10折交叉验证法将使学习方法运行100次。总而言之，当评估一个学习方案的性能时，所进行的任何参数调整过程都应被看作是训练过程的一部分。

当把机器学习技术应用于实际的数据挖掘问题时，还有其他一些重要程序可以大大提高成功率，这正是本章的主题。它们形成了一种（操纵）数据的技术，将输入数据设计成一种能适合所选学习方案的形式，将输出模型设计得更为有效。你可以把它们看成是能应用于实际的数据挖掘问题以提高成功几率的一些诀窍。有时奏效，有时无效。根据目前的技术发展水平来看，很难预言它们是否有用。在这种以尝试和误差率作为最为可靠的指导的领域中，特别重要的恐怕就是灵活运用并且理解这些诀窍了。

先来考察对输入进行修改，使之更适合于学习方案的四种不同方法：属性选择、属性离散、数据转换和数据清理。首先来考虑属性选择。在许多实际情况中，有太多的属性需要学 286

习方案进行处理，其中部分或绝大多数属性是明显无关或重复的。因此，需要对数据进行预处理，从中挑选出一个属性子集运用于学习中。当然，学习方案本身也会进行属性选择，忽略无关的和重复的属性，然而在实际操作中进行预选，常常能提高它们的性能表现。例如，试验显示，增加无用的属性会导致诸如决策树和规则、线性回归、基于实例学习器以及聚类等学习方法的性能表现变糟。

如果任务牵涉到数值属性而所选的学习方案只能处理分类问题时，数值属性的离散化就是绝对必要的。如果将属性进行预先离散处理，能够处理数值属性的方案，经常能获得更好的结果或工作更加迅速。反过来，也有需要将类别属性表示为数字形式的（虽然不常见），我们也将讨论适用于这种情况的技术。

数据转换包含很多技术。曾经在第2章的关系数据以及第6章的支持向量机中遇到过一种转换技术，它增加新的合成属性，目的是要将现有的信息表现成一种适合学习方案的形式。还有那些不那么依赖于具体数据挖掘问题的、更为通用的技术，包括主分量分析和随机投影。

不清洁的数据困扰着数据挖掘工作。在第2章中曾强调认识数据的必要性，了解所有不同属性的含义、对属性进行编码所使用的惯例、残缺值及重复数据的意义、测量干扰、排版印刷错误以及系统误差，甚至是蓄意的。各种简单的可视化经常有助于解决这类问题。然而，也有一些自动清理数据、侦察例外以及发现异常的方法。

在学习了怎样修改输入之后，转向设计机器学习方案的输出结果问题。特别是要考察把从数据所得的不同模型组合起来的技术。其中蕴涵着一些出乎意料的东西。比如，取得训练数据后，从中获取几个不同的训练集，从每个训练集上学习一个模型，再将各个结果模型组合起来，这种方法经常能获得优势！有可能将一个相对弱势的学习方法转变成一个极其强大的方法（我们将从某种精确意义上来进行解释）。另外，如果现存几种学习方法可供选择，使用所有的方法然后将结果组合起来，而不是（使用交叉验证）从中选择最佳方案，这样也许会更有优势。最后，将多类学习问题当作二类问题来建模的标准方法，通过一种简单而灵活的技术能得到改善。
287

许多结果都是相当反直觉的，至少第一眼看上去是这样的。同时采纳多种不同的模型怎么可能是个好建议呢？你怎么可能比选用性能表现最佳的模型做得更好呢？难道所有这些都是要与主张简单化的Occam剃刀理论背道而驰吗？怎么可能通过组合普通的模型获得一流的性能表现？但是考虑由多人组成的委员会做出的决定，通常能比由单个专家所做出的更为明智。回顾一下伊壁鸠鲁的观点，面对各种不同的解释时，必须保留所有的。设想有一群专家，虽然没有一个人是能够完全胜任的，但每个人都在某个特定的领域内有过人之处。在努力理解这些方法是如何工作的过程中，研究者们揭示了能获得更好改善的各种联系和环节。

另一个不寻常的事实是分类性能常可以通过扩充利用大量没有类标签的数据，换句话说就是类值未知的数据而得到改善。与其说这像是常识，还不如说这更像是一条往上游流淌的河流或是一台永动机。但如果这确实是真的，我们将在第7.6节中说明，它将具有很重要的实践意义，因为在许多情况下有类标的数据很少，而没有类标的数据却很多。请继续往下读，准备好领受更多的惊讶。

7.1 属性选择

多数机器学习算法都是要学习哪些属性最适合用于做决策。例如，决策树是在每个节点

挑选最有希望成功的属性进行分裂的，从理论上讲决不选择无关的或是无用的属性。属性越多，从理论上看，需要更强的识别能力，不能更差。“理论上和实践中有何差别？”这是个老问题。答案是“从理论上看，理论和实践没有差别，但在实践中有差别。”这里也一样，在实践中，往数据集里添加无关或干扰属性，经常使机器学习系统“糊涂”。

决策树学习器（C4.5）的实验显示，往标准数据集中添加一个随机的二值属性，属性值由抛掷无偏硬币产生，这会影响分类性能，导致性能变差（在这种测试情形中下降了5%至10%）。变差的原因是在树的某些节点处，这个无关的属性被不可避免地选择为决定分支的属性，导致使用测试数据测试时产生随机误差。决策树学习的设计是非常巧妙的，能在每个节点挑选最适合的属性进行分裂，怎么会发生这种情形的呢？原因也很微妙。随着程序渐渐向树的下层运行，能对属性选择决定有帮助的数据变得越来越少。在某个节点，数据极少，随机属性碰巧看起来较好。由于每层的节点数量是随层数按指数级增加的，这个无赖属性在某处看起 288
来较好的几率也随着树的深度成倍增加。真正的问题是树总是会到达某个深度，那里只存在少量的数据可用于属性选择。即使数据集较大，也不能避免这个问题，只是树可能更深而已。

分治决策树学习器和割治规则学习器都存在这个问题，因为作为判断基础的数据数量一直在减少。基于实例的学习器非常容易受无关属性的影响，因为它们始终是在局部近邻范围内工作，每个决策只考虑少数几个实例便做出。实际上，基于实例学习法要达到某个预设的性能水平，所需的训练实例数量是随无关属性的数量按指数级增加的。相反，朴素贝叶斯不会将实例空间分割成碎片并忽略无关属性。它假设所有属性都是相互独立的，这个假设正适合（处理）随机“干扰”属性。但也正是由于这个假设，朴素贝叶斯在另一方面却付出了重大代价，由于加入重复属性使其工作受挫。

无关的干扰会使决策树和规则学习器的性能降级是令人吃惊的。更令人惊讶的是相关属性也会是有害的。例如，假设在一个二类数据集中加入一个新的属性，大多数情形（65%）这个属性值与预测的类值是相同的，而其余情形则是相反的，这两种情形在数据集中是随机分布的。使用标准数据集进行试验的结果显示，这会造成分类正确率下降（在这个试验情形下为1%至5%）。问题出在新属性在决策树的上层便被（自然）选中用以分裂。存在与下层节点处的是分割了的实例集，以致其他属性选择只能基于稀疏的数据。

由于无关属性在多数机器学习方案中存在负面影响，通常在学习之前先进行属性选择，只保留一些最为相关的属性，而将其他属性都去除。选择相关属性最好的方法是人工选择，它是基于对学习问题的深入理解以及属性的真正含义而做出的选择。然而，自动方法也是很有用的。通过去除不适当的属性以降低数据的维数，能改善学习算法的性能。还能提高速度，即使属性选择可能会带来很多计算。更重要的是，维数降低能形成一个更为紧凑、更易理解的目标概念表达方式，使用户的注意力集中在最为相关的变量上。 289

7.1.1 独立于方案的选择

选择一个好的属性子集，有两种根本不同的方法。一种是根据数据的普遍特性做出一个独立评估；另一种是采用将要用于最终机器学习的算法来评估子集。第一种称为过滤（filter）方法，因为它是要在学习开始之前，先过滤属性集产生一个最有前途的属性子集。第二种称为包装（wrapper）方法，因为学习方法被包裹在选择过程中。如果存在一个好方法能判定何时一个属性与类选择是相关的，那么要对某个属性子集做独立评估便很容易了。虽然人们提

出了几种不同的建议，但是目前还没有能被一致接纳的“相关性”量度。

在属性选择中，一种简单的独立于方案的方法便是使用足够的属性来分割实例空间，使所有的训练实例分隔开来。例如，如果只考虑一个或两个属性，通常会有几个实例都含有相同属性值组合；而另一个极端情形是考虑整个属性集便能区分每个实例，因此不存在所有属性值都是相同的两个实例。（虽说不是必然的，但是数据集有时会存在具有相同属性值的实例却分属于不同类的情形。）直觉让我们选择能区分所有实例的最小的属性集。这可以通过穷举搜索很容易地找到，虽然计算相当耗费。不幸的是，这个源于训练数据上的、具有强烈偏差倾向的属性集，它的一致性并不能得到保证，而且会导致过度拟合，算法可能会卷入不必要的工作，用以修复只是由于某些干扰数据引起的不一致性。

机器学习算法可用于属性选择。例如，可先在整个数据集上应用决策树算法，然后选择那些在决策树中真正用到的属性。如果下一步只是要建另一个树，那么这个属性选择不会产生任何效果；然而这个属性选择将对不同的学习算法产生影响。比方说，众所周知，最近邻算法很容易受无关属性的影响，它便可以通过先建一个决策树过滤属性而使性能得到提高。最终所得的最近邻结果性能比用于过滤的决策树的性能更好。再举一个例子，在第4章中提到的简单的单规则（1R）方案，通过对不同属性分支的效果进行评估，可用于决策树学习前的属性选择（然而像1R 这样的基于误差的方法也许不是属性排序的理想选择，我们将在下面讨论有指导离散所相关的问题时看到这点）。通常只用排在前两、三位的属性构建的决策树也能达到同样好的性能，而且这个树更容易理解。在这个方法中，用户要决定使用多少属性来构建这个决策树。

另一个可行的算法是建一个线性模型，比如一个线性支持向量机，根据系数的大小来进行属性排序。一种更为精密复杂的方法是将这个算法重复运行。先建一个模型，根据系数进行属性排序，将排在最高位的属性移除，然后重复这个过程直到所有属性都被移除。这种递归属性消除（recursive feature elimination）的方法应用于某些数据集上（譬如识别重要基因
290 用于癌症分类）与只用单个模型进行属性排序的方法相比较，能获得更好的结果。对于这两种方法来说，保证属性具有相同的衡量尺度是很重要的，否则这些系数是不可比的。注意这些技术只是用于产生一个属性排序，还必须应用其他技术来决定适当的属性数量。

还可以使用基于实例的学习方法来进行属性选择。从训练集中随机抽样，检查近邻的同类和不同类实例记录，即“近邻击中”（near hits）和“近邻错失”（near miss）。如果近邻击中的某个属性有不同值，这个属性看来似乎是不相关的，那么它的权值就要降低。另一方面，如果近邻错失的某个属性有不同值，这个属性看来是相关的，它的权值就要增加。这是基于实例学习的属性加权的标准程序，在6.4节中曾描述过。在这个操作过程重复多次后，便进行选择操作：只选择权值为正数的属性。正如同标准的递增基于实例的学习，由于样本排列的次序不同，每次重复过程都会得到不同的结果。这点可以通过使用所有的训练实例并考虑每个实例所有的近邻击中和近邻错失来避免。

一个更为严重的缺点是无法探察出重复属性，因为它与另一个属性是相关的。极端的情形是，两个相同的属性被同样处理，不是都被选择便是都不被选择。有人提议了一种修正方法似乎有助于这个问题的解决，即在计算最近邻击中和错失时考虑当前的属性权值。

既消除无关属性也消除重复属性的另一种方法便是选择一个属性子集，属性各自与类属性有较大关联但几乎没有内部关联。两个名词性属性A和B之间的关系可用对称不定性

(symmetric uncertainty) 来衡量: 291

$$U(A,B)=2\frac{H(A)+H(B)-H(A,B)}{H(A)+H(B)}$$

H是4.3节中讲述的熵函数。熵是以每个属性值的概率为基础的；$H(A,B)$是A和B的联合熵，由A和B的所有组合值的联合概率计算出来的。对称不定性总是在0和1之间。基于相关性(correlation-based)的属性选择决定一个属性集的优良性是采用

$$\sum_j U(A_j,C)\Big/\sqrt{\sum_i\sum_j U(A_i,A_j)}$$

这里C是类属性，i和j包括属性集里的所有属性。假设子集中所有m个属性与类属性及另一属性密切相关，分子就为m，分母为$\sqrt{m^2}$，亦即m，因此得到最大值1（最小值为0）。显然这不理想，因为我们要避免重复属性，而这个属性集中的任何子集的衡量值都会是1。当使用这个标准来寻找一个好的属性子集时，便是要选择最小的子集来打破平局。

7.1.2 搜索属性空间

属性选择的大多数方法都要牵涉到在属性空间搜索最有可能做出最好类预测的属性子集。图7-1展示了我们最熟悉不过的天气数据的属性空间。可能的属性子集数目随属性数量的增加呈指数增长，使得穷举搜索不切实际，它只适合最简单的问题。

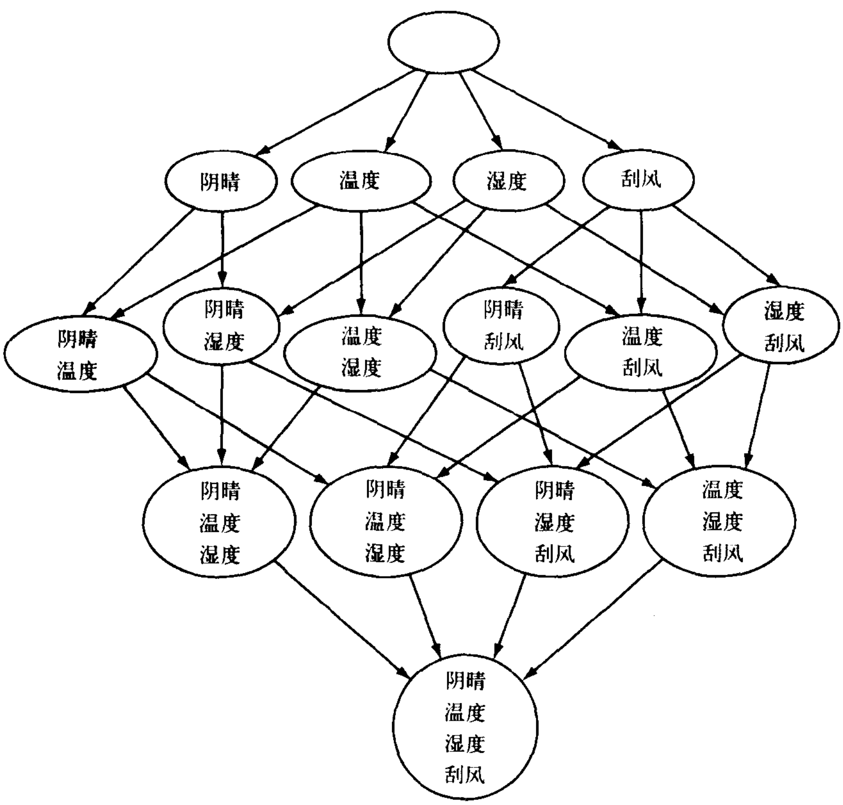


图7-1 天气数据集的属性空间

基本上，搜索方向是两个方向中的一个，即图中从上往下或是从下往上。在每个阶段，通过增加或删除一个属性，来改变目前的属性子集。朝下的方向，开始时是不含任何属性，然后每次增加一个，称为正向选择。朝上的方向，开始时包含了所有属性，然后每次减少一个，称为反向消除。

在正向选择时，每个当前子集没有包含的属性被暂时加入，然后对结果子集的属性进行评估，譬如使用下节中要描述的交叉验证。评估产生一个数字结果用以衡量子集的期望性能。通过这个方法对依次添加每个属性所产生的结果进行定量，选择其中最好的，然后继续。如果向目前的子集中添加任何一个属性都不能有改善时，即终止搜索。这是一个标准的贪心搜索程序，能保证找到一个局部的，不必是全局的、最好的属性集。反向消除操作采用完全类似的模式。在这两种情形中，对于较小的属性集经常引用一个微小偏差。在正向选择中，如果要继续搜索，评估值的增加必须要超出某个预先设定的最小增量。对于反向消除也采用类似的方法。

还存在一些更为精细复杂的搜索方法。正向选择和反向消除法可以结合成双向搜索，同样一开始可以包含所有属性或者不含任何属性。最佳优先搜索（best-first search）不是在性能开始下降时停止搜索，而是保存到目前为止已经评估过的所有属性子集清单，并按照性能衡
292~293 量好坏排序的方法，因此它可以重访先前的配置。只要时间允许它将搜索整个空间，除非是采用某种停止标准。限定范围搜索（beam search）也很类似，只是在每个阶段截取属性子集清单，因此它只含有固定数目的束宽（beam width）中最有希望的候选对象。遗传算法（genetic algorithm）搜索程序松散地基于自然选择原理，使用对当前的候选子集的随机扰动，而"进化出"好的属性子集。

7.1.3　特定方案选择

采用特定方案（scheme-specific）选择的属性子集性能表现是根据学习方案仅仅使用这些属性进行分类的性能来衡量的。给定一个属性子集，正确率是采用5.3节中所述的交叉验证法来评估的。当然，其他评估方法，如在一个旁置集上的性能表现（5.3节）或者使用自引导估计器（5.4节）也同样适用。

整个属性选择过程是计算密集型的。如果每个评估都采用10折交叉验证，学习过程要执行10次。对于k个属性，正向选择或是反向消除法评估时间要乘以一个因数，这个因数最大可达k^2，对于更为复杂的搜索，这个因数远大于此，对于穷举算法要检验2^k种可能的属性子集，这个因数可达2^k。

已被证实（这个方法）在许多数据集上都获得好的结果。一般来说，反向消除法较正向选择法生成的属性集较大，分类正确率较高。原因是性能衡量只是一个估计，单一的优化估计导致这两种搜索过程过早停止，反向消除的结果属性还过多，而正向选择的结果属性还不够。如果重点是要理解所涉及的决策结构，那么正向选择是很有用的，因为它经常是减少了属性数目而对分类正确率的影响却又很小。实践似乎显示复杂的搜索技术并不总是适当的，虽然它们在某些情况下能产生很好的结果。

加速搜索过程的一种方法是一旦发现不太可能产生比另一个候选子集更高的正确率，便立即停止对这个属性子集的评估。这项工作是统计学上的成对显著性测试（significance test）工作，它基于这个属性子集所生成的分类器和所有基于其他子集的候选分类器之间进行。两

个分类器对某个测试实例分类的性能差异可表示为-1、0或1，这是根据第一个分类器较第二个更差、相同或是更好而决定。可以将成对的t测试（t-test）(5.5节)应用于整个测试集上所得 294
的这些数字，从而有效地将每个实例的（分类）结果当成是一个性能差异的独立估计来处理。一旦能看出一个分类器明显比另一个差时，当然也许不会发生，则立即停止交叉验证。我们可能希望更积极地丢弃分类器，这可通过修改t测试，计算一个分类器比另一个更好的概率至少要达到某个用户指定的小阈值。如果这个概率非常小，我们可以丢弃前面一个分类器，因为它比后者明显好的可能性很小。

这个方法称为竞赛搜索（race search），实施时可以采用不同的搜索策略。当采用正向选择策略时，所有可能的单个属性添加同时进行竞赛，去除那些表现不够好的。在采用反向消除策略时，所有可能的单个属性去除同时进行竞赛。模式（Schemata）搜索是一种为竞赛特制的更复杂的方法，它要进行一系列的迭代竞赛，每次都要决定是否要包含某个具体属性。竞赛中其他属性在评估中包含或不包含是随机的。一旦竞赛出现一个明显优胜者，便开始下一轮的迭代竞赛，将优胜者作为起始点。另一种搜索策略是先将属性排序，例如利用它们的信息增益(假设是离散的)，然后对排序进行竞赛。这时竞赛包括不含任何属性、排列第一的属性、排列前两位的属性、前三位属性，依次类推。

无论采取何种方法，特定方案属性选择决不是始终都能获得性能提高的。由于过程的复杂程度，在属性选择循环中包含了最终机器学习算法的反馈效果使复杂度大大增加，非常难预测在何种条件下是有价值的。正如在许多机器学习情况下，使用自己的数据反复试验是最终的仲裁者。

有一种分类器，它的特定方案属性选择是它学习过程的一个重要部分：决策表（decision table）。正如在3.1节中所提到的，学习决策表的全部问题在于选择合适的属性。通常是通过对不同属性子集进行交叉验证，选择表现最好的属性子集。所幸的是，留一（leave-one-out）交叉验证对于这种分类器来说是非常廉价的。由于添加或删除实例并不改变表的结构，从由训练数据产生的决策表获得交叉验证误差，仅仅是要处理类的累计以及相关的表中每个项目。属性空间的搜索通常是采用最佳优先的方法，因为这个策略与其他策略，如正向选择，相比较在局部最大点被卡可能性更小。

让我们用一个成功的故事来结束讨论。有一种学习方法，采用了简单的特定方案属性选择方法并具有较好表现，那便是朴素贝叶斯法。虽然这个方法对于随机属性处理得相当好，但当属性之间存在依赖关系时，特别是在加入了重复属性时，它存在误导的倾向。然而，使用正向选择算法，当有重复属性加入时，与反向消除法相比，此法具有更强的侦察力，并采用一个非常简单、几乎是“幼稚”的量度来决定属性子集的质量，即采用学习算法在训练数 295
据集上的性能表现。在第5章中曾强调，在训练集上的性能表现绝对不是测试集性能的可靠指示器。然而，试验表明对朴素贝叶斯法所做的如此简单的改进，在那些原本性能表现不如其他基于树或基于规则分类器的标准数据集上，现在却能获得显著的性能提高，而对于那些原本就表现较好的数据集也没有任何负面影响。选择性的朴素贝叶斯法，这种学习方法正如其名称，是一种在实践中可靠的、性能良好的机器学习技术。

7.2 离散数值属性

一般数据集，数值属性必须先被“离散”成一些不同的值域。即使是能够处理数值属性的学习算法，有时处理方法也并不完全令人满意。统计聚类方法常假设数值属性呈正态分布，这在实践中时常是不太合理的假设。朴素贝叶斯分类器用于处理数值属性的标准延伸法，也采用同样的假设。虽然大多数的决策树和决策规则学习器可以处理数值属性，但是当出现数值属性时，由于需要重复对属性值进行排序，有些分类器工作变得相当慢。由于上述种种原因的存在产生了一个问题，在进行学习之前将数值属性离散成不同的值域，该采取什么样的方法才好？

我们先前已经看到一些离散数值属性的方法。在第4章中讨论的单规则（1R）学习方案采用了一种简单而有效的技术，根据属性值将实例进行排序，在类出现变化时设定属性值值域，除了每个值域中所含的多数类实例必须达到某个最小数目（6个），这意味着每个值域中所含的类值是混合的。这是一种在开始学习之前能应用于所有连续属性的“全局”化的离散方法。

另一方面，决策树学习器是在局部进行数值属性处理，在树的每个节点处，当决定是否值得分支时对属性进行检查，并且只在这个节点处决定连续属性的最佳分裂点。虽然在第6章中讨论的建树方法只考虑将连续属性一分为二，可以想象在这个节点做完全的离散从而对数
296 值属性进行多路分裂。采用局部方法和全局方法的优缺点很清楚。局部离散是为适应每个节点的实际情况，同一个属性在树的不同位置，只要看起来适当，会产生不同的离散结果。然而，随着树的深度增长，这个决定是基于较少的数据而做出的，会影响到它的可靠性。如果树在修剪之前是一直往下扩展直到单个实例的叶节点，如同采用普通的反向修剪法技术，很明显许多离散决定是基于非常不充足的数据而做出的。

在应用学习方法之前采用全局离散，有两种可能方法能将离散了的数据呈现给学习器。最明显的方法便是把离散了的属性当作名词性属性来处理，每个离散区间用名词性属性的一个值来代表。然而，由于离散了的属性是从数值属性演变而来，它的值是有序的，把它当作名词性属性处理便丢弃了它潜在的颇有价值的排序信息。当然，如果学习方案能够直接处理有序属性，解决方法显而易见，将每个离散了的属性申报为“有序”型的属性。

如果学习方法不能处理有序属性，仍然有一个简单的方法可用来利用排序信息，在使用学习方法之前，将每个离散了的属性转换成一组二值属性。假设离散了的属性有k个值，将它转换为$k-1$个二值属性，每当数据呈现为第i个离散了的属性值时，便将前$i-1$个二值属性设定为false，其余的都设为true。换句话说，第$(i-1)$个二值属性代表了离散属性是否小于i。如果决策树学习器要分裂这个属性，它可利用这个编码中所隐含排序的信息。注意这个转换是独立于所应用的具体离散方法的，它只是用一组二值属性来对一个有序属性进行编码。

7.2.1 无指导离散

离散问题有两种基本方法。一种是在训练集实例类未知的情况下，对每个属性量化，即所谓的无指导离散（unsupervised discretization）。另一种是在离散时要考虑类属性，即有指导离散（supervised discretization）。前者只有在处理类未知或类不存在的聚类问题时，才有可能碰到。

离散数值属性的直观方法便是将值域分隔成几个预先设定的等区间：一个固定的独立于数据的尺度。通常是在收集了数据后进行。但是，如同其他无指导离散法一样，它存在某些
297 风险，由于使用的等级过于粗糙而破坏了在学习阶段中可能有用的差别，或者不幸选择了将

许多不同类的实例不必要地混在一起的分隔边界。

等区间装箱（equal-interval binning）经常造成实例分布非常不均匀：有些箱中包含许多实例，而有的却一个也没有。这样会严重削弱属性帮助构建较好决策结构的能力。通常，允许有不同大小的区间存在，从而使每个区间内的训练实例数量相等会更好些。这种方法称为等频区间装箱法（equal-frequency binning），根据这个轴上的实例分布将属性值域分隔成几个预先设定的区间，有时称为直方图均衡化（histogram equalization）。因为观察结果区间内容的柱状图，会看到它是完全平直的。如果把区间数目作为一种资源，这个方法最适合不过了。

然而，等频区间装箱仍然不注意实例的类属性，这将导致不良的分界。例如，如果一个区间中所有实例都属一个类，而下一个更高的区间中除了第一个实例仍属于先前的类，其余的实例都属另一个类，理所当然应尊重类特性分隔，将第一个实例包括到前一个区间中，牺牲等频特性以保全同种特性是有意义的。有指导的离散，即在离散过程中考虑类特性当然有优势。然而，人们发现等频装箱能带来很好的结果，至少是在与朴素贝叶斯学习法一同应用时，此时区间个数依数据而被设定为实例总数目的平方根。这个方法称为均衡k区间离散（proportional k-interval discretization）。

7.2.2 基于熵的离散

由于决策树形成过程中所采用的分裂数值属性的方法在实践中效果较好，看来采用递归分裂区间直至终止时间，从而将其延伸成更为普及的离散法不失为一个良策。在第6章中我们看到了该如何将实例按照属性值排序，如何在每个可能的分裂节点处考虑分裂结果的信息增益。进行属性离散时，第一次分裂一旦决定，分裂过程可以在上部值域或下部值域重复进行，采取递归依次类推。

为了看清在实践中是如何工作的，再来回访一下（6.1节）离散天气数据中温度属性的例子，属性值为 298

64	65	68	69	70	71	72	75	80	81	83	85
yes	no	yes	yes	yes	no	no yes	yes yes	no	yes	yes	no

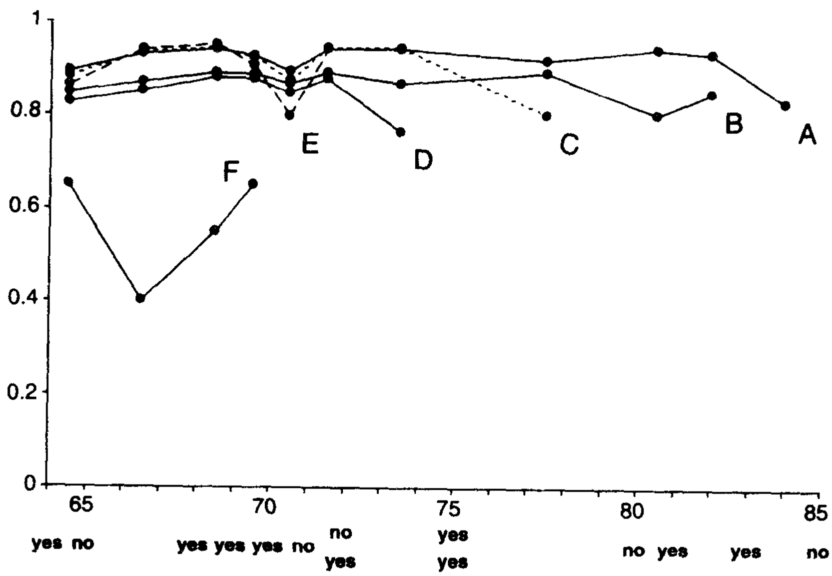
（重复值已经被叠在一起了。）11处可能的分裂点的信息增量按照常规方法计算。例如，温度<71.5的测试，将值域分裂为包含4个yes、2个no对应5个yes、3个no，其信息值为：

$$\text{info}([4,2],[5,3]) = (6/14) \times \text{info}([4,2]) + (8/14) \times \text{info}([5,3]) = 0.939\ \text{位}$$

这个值代表了给出此分裂后要详述各个yes和no值所需的信息总量。我们要寻找一种离散使子区间尽可能地纯正，因此，要选择信息值最小的分裂点（这等同于要在信息增益最大处分裂，信息增益定义为未分裂与分裂后的信息值差值）。如前所述，将数值阈值置于概念分界区域中间位置。

图7-2中标记为A的曲线显示了第一个阶段每个可能的分裂点的信息值。最好的分裂即信息值最小处在温度为84处（0.827 位），它只是将排列在最后的、类值为no的实例从原先的序列中分离出来。在水平轴底下标出了实例的类属性以便说明。在温度的低值域64至83，再次应用这个算法产生图中标记为B的曲线。这次最小值在80.5（0.800位），将接下来的两个同属yes类的实例分离开。再次在低值域上应用算法，现在是从64到80，产生标记为C的曲线（图中用点线表示与其他曲线区别开来）。最小值在77.5 处（0.801位），又分离出一个类值为no的

实例。曲线D最小值在73.5 处（0.764位），分离出两个类值为yes的实例。曲线E （再次使用虚线，只是为了易于分辨），温度值域从64到72，最小值在70.5 处（0.796位），它分离出两个类值为no和一个类值为yes的实例。最后曲线F，值域从64到70，最小值在66.5 处（0.4位）。



299

图7-2　采用熵方法离散温度属性

最终的温度属性离散结果如图7-3所示。递归只发生在每次分裂后的前一个区间上，仅仅是这个例子出现的结果。一般来说，上部和下部区间都将被进一步分裂。每次分裂下方所示的（字母）是图7-2中对应与该次分裂的曲线标记，底部是分裂点真正的分裂值。

64	65	68	69	70	71	72	75	80	81	83	85
yes	no	yes	yes	yes	no	no yes	yes yes	no	yes	yes	no

分裂	F	E	D	C	B	A
分裂值	66.5	70.5	73.5	77.5	80.5	84

图7-3　温度属性离散结果

从理论上来看，最小的信息值绝对不会出现在两个同属一个类的实例之间。这点引入了一个有用的优化，只需考虑发生在不同类实例间的潜在分裂。注意如果区间的类标签是基于区间的多数类而定的，便不能保证相邻的区间具有不同的类标签。你也许会考虑将多数类相同的区间进行合并（例如，图7-3中的前两个区间），但下面我们将看到（7.2节），通常这样做并不好。

剩下要考虑的问题就是停止标准了。在温度例子中，大多数经确认的区间都是"纯粹的"，即区间内所有实例的类值都相同，很清楚没有必要再去分裂这样的区间。（最后一个区间，我

们默认不再分裂，还有70.5到73.5这个区间，是两个例外。）然而，一般情况下问题并非是如此直截了当的。 300

要终止基于熵分裂的离散程序的一个好方法是利用曾在第5章中遇到的MDL 原理。根据那个原理，要使“理论”规模加上给定理论详述所有数据所需的信息量达到最小值。在这个情况下，如果进行分裂，“理论”是分裂点，要比较分裂和不分裂理论的情形。在这两种情况下，假设实例已知，但它们的类标签未知。如果不做分裂，类别传输可以通过将每个实例的标签进行编码来实现。如果进行分裂，先将分裂点编码（$\log_2[N-1]$位，这里N是实例数量），然后是小于这个分裂点的类，再是大于这个分裂点的类。可以想象，如果这是一个好的分裂点，譬如说，所有小于这点的类值都是yes，所有大于这点的类值都是no，那么分裂便会带来相当大的益处。如果yes和no的实例数量相等，不采用分裂时，每个实例耗费1位，而使用分裂时耗费几乎是0，不完全等于0。因为类值以及分裂本身都要被编码，但这个惩罚是被分摊到所有的实例上的。在这种情形下，如果有许多实例，由分裂所节省的信息量将远超出必须对分裂点进行编码的惩罚。

在5.9节中曾强调，应用MDL原理时，困难会随着对问题细节的深入而出现。在相对直接的离散情形中，虽然也不简单，但还是较易处理的。信息总量可以在某些合理的假设条件下获得。这里我们不再深入讨论，结论是如果这个分裂所带来的信息增益超出某个定值时，在某处进行分裂还是值得的，这个定值是由实例数量N、类别数量k、实例集E的熵、每个子区间内实例集E_1和E_2的熵以及每个子区间内的类别数量k_1和k_2所决定的：

$$\text{增益} > \frac{\log_2(N-1)}{N} + \frac{\log_2(3^k-2) - kE + k_1E_1 + k_2E_2}{N}$$

第一个部分是描述分裂点所需的信息，第二部分是传输哪些类别分别对应与前面和后面的子区间所需的一个修正量。

当应用在温度例子上时，这个标准阻止任何分裂。第一次分裂只分离出了最后一个实例，正如你所能想象的，传输这些类别时几乎没有获得什么真正的信息。实际上，MDL标准不会产生任何只含一个实例的区间。离散温度属性的失败，取消了温度属性在最终的决策树结构中担当任何角色，因为赋予所有实例的离散值都会是一样的。在这种情形下，这是完全正确的：对天气数据来说，温度属性在好的决策树或决策规则中都不会出现。实际上，离散失败等价于属性选择的效果。 301

7.2.3 其他离散方法

采用基于熵的方法并应用MDL停止标准是有指导离散的最好的通用技术之一。人们还研究了许多其他方法。例如，替代原先自上而下，递归分裂区间直至满足某个停止标准的程序，你也可以由下往上，先将每个实例置于各自的区间，然后考虑是否合并相邻区间。可以应用统计标准来观察哪两个是最佳合并区间，如果统计结果超出某个事先设定的置信水准便将它们合并，重复此过程直到不再有能通过测试的潜在合并。χ^2测试很合适，并运用于此。更为复杂的技术用于自动决定合适的（置信）水准来替代预先设置重要性阈值。

一个相当不同的方法是在对训练实例的类进行预测时，累计离散所带来的误差数量，假设每个区间接受多数类。例如，前面所述的单规则（1R）方法是基于误差的，它更注重于误差而非熵 。然而，根据误差累计所获得的最佳离散，往往使用尽可能大的区间个数而得到，

必须预先限定区间个数来避免这种退化情形。你也许会问，要将某个属性离散成k个区间并使其误差数目达到最小，什么才是最好的方法？

要寻找能将一个属性分裂成k个区间，并使与k成指数关系的误差累计达最小化的最佳方案的穷举方法是不现实的。但是，基于动态规划（dynamic programming）理念有一些非常有效的方案。动态规划不仅应用于误差累计衡量，而且应用于任何给定的不纯函数，能找到将N个实例分裂成k个区间，并使不纯度最小化的时间与kN^2成比例。这给出了一种方法用以寻找最佳基于熵的离散，使先前讲述的递归基于熵的离散方法得到潜在改善（但在实践中这种改善却是可忽略的）。基于误差离散的消息更加好一点，因为有一种方法可使误差累计最小化的时间与N成线性关系。

7.2.4 基于熵和基于误差的离散

既然优化的离散运行非常快，为什么不采用基于误差的离散？答案是因为基于误差的离散
302 散有一个严重的缺点，它不让相邻区间有相同的类标签（如图7-3中的前两个）。原因是合并这样两个区间不影响误差累计，却能释放一个区间可用于（离散）别处来降低误差累计。

为什么有人要生成两个相同类标的相邻区间呢？下面的例子是最好的解释。图7-4展示了一个简单的二类问题的实例空间，它含有两个数值属性，属性值范围从0到1。如果第一个属性（$a1$）值小于0.3，或者小于0.7且第二个属性（$a2$）值小于0.5，那么实例属于一类（图中圆点）；否则，实例便是属于另一类（图中三角）。图7-4中的数据便是根据这些规则人工生成的。

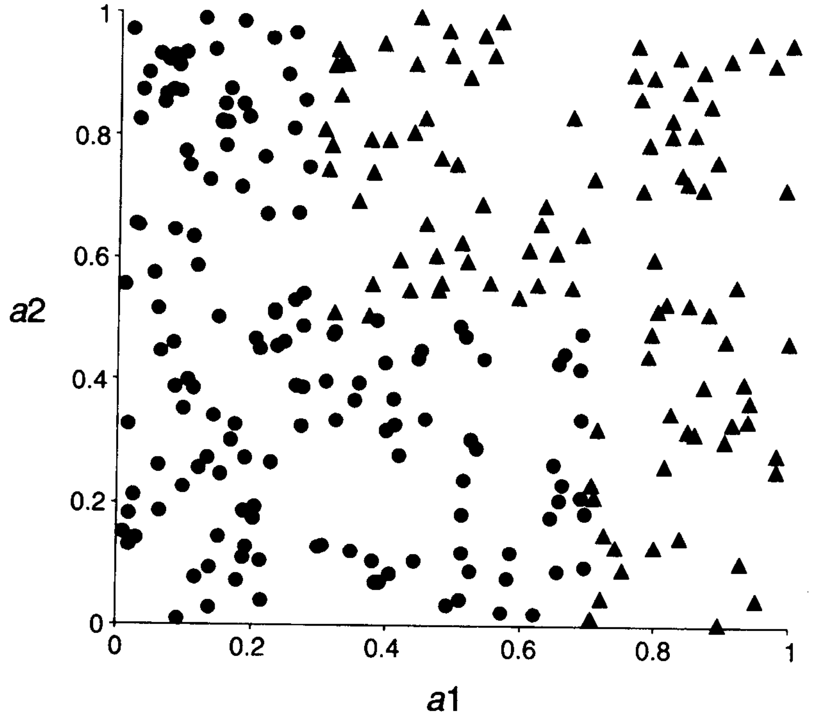


图7-4　含两个类别、两个属性问题的类分布

现在假设要将两个属性进行属性离散，以便从离散了的属性中进行分类学习。最好的离散分裂就是将$a1$分裂成3个区间（0到0.3，0.3到0.7，0.7到 1.0)，将$a2$分裂成2个区间（0到0.5，0.5到1.0)。给定这些名词性属性，要学习怎样用简单的决策树或规则算法分类就很容易了。分裂$a2$没有问题。对于$a1$来说，第一个和最后一个区间具有相反的类标签（分别是圆点和三角)。第二个类标签是从0.3到0.7这个区间上出现较多的那个类别（对于图7-4中的数据来说实际上是圆点)。不管是哪种类，这个标签肯定会与一个相邻的标签相同，无论中间区域的类概率是多少，这点都是成立的。因此，这种离散是任何误差累计最小化的方法都不会得到的，因为这类方法不允许相邻区间产生同样的标签。 303

重点是当$a1$的属性值越过0.3这个界线时，改变的不是多数类而是类分布。多数类仍然是圆点，但类分布发生了显著变化，从以前的100%到刚过50%。当越过0.7界线后，类分布再次变化，从50%降低到0%。即使多数类不变化，基于熵的离散方法对于类分布变化也敏感。而基于误差的方法却不敏感。

7.2.5 离散属性转换成数值属性

有一个和离散相反的问题。有些学习算法特别是基于实例的最近邻法和牵涉到回归的数值预测技术只处理数值属性。怎样将它们扩展应用于名词性属性呢？

如4.7节中所描述的基于实例的学习可以通过定义两个名词性属性值之间的“距离”把离散属性当作数值属性来处理，相同值距离为0，不同值距离为1，不管实际数值是多少。不用修改距离函数，而是使用一种属性转换来实现，将一个含有k个属性值的名词性属性用k个合成二值属性来替代，每个（二值属性）对应一个（名词性属性）值，指示这个属性是否有这个属性值。如果属性权值相同，(这些属性)在距离函数上的影响是相同的。距离对于属性值是不敏感的，因为只有“相同”或“不同”两种信息被编码了，而不是与各种可能的属性值相关联的差别度。如果属性带有反映它们相对重要性的权值，便能获得更多的细微差别（信息)。

如果属性值可以排序，会带来更多的机会。对于一个数值预测问题，对应于每个名词性属性值的平均类值可以从训练实例中计算得到，并可用它们来定出一个序列。这种技术是为6.5节中的模型树而引入的。(对于分类问题，很难找到类似的方法对属性值进行排序。）很明显，一个排序的名词性属性可以用一个整数来代替，这不仅暗示一个序列，还暗示一个属性值的衡量尺度。可以通过为一个含k个值的名词性属性建立$k-1$个合成二值属性（7.2节所描述的方法）来避免牵涉衡量尺度。这种编码方法仍然表示不同属性值之间的排序，相邻的值只会有一个二值属性不同，而间隔的值会有几个二值属性不相同，但这时属性值之间不是等距离的。 304

7.3 一些有用的转换

资源丰富的数据挖掘者拥有满载挖掘技术的工具箱，譬如用于数据转换的离散技术。如同在2.4节中所强调的，数据挖掘从来就不是提取一个数据集，将学习算法应用于数据上那么简单的事情。每个问题都不相同。你必须对数据进行思考，琢磨它的意义，然后从不同的角度来检验，具有独创性地找到一个合适的观点。用不同的方法对数据进行转换可以助你拥有一个好的开端。

你不必亲自去实现这些技术来武装自己的工具箱。用于数据挖掘的精细而复杂的操作环

境，为你提供了广泛的有用工具，如本书第二部分所介绍的就是其中的一种。你不必详细了解它们是怎么实现的。你所要了解的是这些工具能干什么，怎样来使用它们。本书第二部分列出并简单介绍了出现在Weka数据挖掘工作台上的所有转换。

数据经常要求对一系列属性进行数学转换。将数学函数应用于现有的属性上定义出新的属性或许会有用。两个日期属性相减可能产生第三个代表年龄的属性，一个受原始属性含义所驱的语义转换的例子。在已知学习算法的一些特性时，也建议运用其他一些转换。如果觉察到一个线性关系牵涉到两个属性A和B，而算法只能进行轴平行（axis-parallel）分裂（如同大多数的决策树和规则学习器），那么将比例A/B定义成一个新的属性。转换不必一定是数学函数，也可能包含一些常识，如一星期所含天数、公休假期或化学原子数量等。转换可以被表示为电子数据表中的操作或用任意计算机程序来实现的功能。或者也可以将几个名词性属性的属性值联合在一起减少成为一个属性，由分别含k_1个值和k_2个值的两个属性形成包含$k_1 \times k_2$个值的单一属性。离散可将一个数值属性转换为名词性属性，我们也看到了怎样进行逆向
305 转换。

另一种转换是在数据集上应用聚类程序，然后定义一个新的属性，对于任何实例来说，新属性的值便是实例所属的聚类类标。你可以在每个实例上应用概率聚类，增加它对于每个聚类的从属概率属性，有多少聚类就添加多少新属性。

有时在数据上添加一些干扰数据会有帮助，可用来检测一个学习算法的强壮度。取一个名词性属性，改变它的属性值的已知比例。通过将关系、属性名称、名词性属性和字符串属性的属性值重新命名使数据混乱，因为使一些敏感数据匿名化经常是有必要的。将实例的次序随机化或通过二次采样（resampling）产生数据集的一个随机样本。按照某个给定比例删除实例，或删除名词性属性为某些值的实例，或者是删除数值属性值高于或低于某个阈值的实例，从而减小实例集。或者在实例集上应用某个分类方法，然后删除被错误分类的实例，以此去除某些例外。

不同类型的输入要求有它们各自的转换。如果能输入稀疏数据文件（见2.4节），也许需要能将数据集转换为非稀疏形式，反之亦然。文本输入和时间序列输入要求有它们各自的特殊转换，将在下面的小节中详述。先来看看将数值属性转换为低维形式的两种普及技术，它在数据挖掘中比较有用。

7.3.1 主分量分析

对于一个含有k个数值型属性的数据集，你可将这些数据想象成在k维空间上密布的点，就像天上的星星、一大群被定格的飞虫、一张纸上的二维点阵图。属性象征着在空间上的坐标轴。但所用的轴（即坐标系统本身）是任意方向的。你可以在纸上放置一个水平方向和一个垂直方向的轴，然后利用这些坐标轴来表示点阵图中的每个点，也可以任意画一条直线代表X 轴，再用一条与它正交的直线代表Y轴。要记录飞虫的位置，可以用传统的定位系统，一条南北向的轴、一条东西向的轴以及一条上下方向的轴。采用其他坐标系统也同样可行。像飞虫这样的生物不懂得东西南北，虽然由于地心引力，它们也许能感知上下方向。至于天上的星星，谁能说出什么是“正确”的坐标系呢？数世纪以来，我们的祖先从以地球为中心的观点转变成以太阳为中心的观点再到纯粹的相对论观点，每次观点的转
306 变都伴随着宗教与科学之间关系的剧变，以及对于人类在上帝所创造的宇宙中所处角色的

痛苦的重新审视。

回到数据集，就像上述例子所示的，没有任何因素可以阻止你将所有的数据点转换到一个不同的坐标系中去。但与上述例子不同的是，在数据挖掘中经常存在一个首选的坐标系，它不是由外在的惯例所决定，而是由数据本身决定的。无论采用何种坐标，数据点在每个方向上都存在一个方差，预示了在这个方向上平均值周围的伸展度。奇怪的是如果将各个方向上的方差相加，然后将数据点转换到一个不同的坐标系统中去并做同样的操作，两种情况下所得的方差总和是相同的。只要坐标系统是正交的，即每个坐标轴与其他坐标轴都成直角，这种关系始终是成立的。

主分量分析（principal components analysis）法的思想是使用一个特殊的、由数据点决定的坐标系、将第一个坐标轴设在数据点方差最大的方向上，从而使这个轴向上的方差最大化。第二个坐标轴与第一个轴正交。在二维空间没有其他选择，它的方向由第一个轴所决定。但在三维空间，它可以是在与第一个轴正交的平面上的任意位置，在更高维的空间甚至有更多的选择，虽然始终限制其必须与第一个轴正交。遵循这个限制，选择沿轴向的方差达到最大值的方向作为第二个轴向。如此继续选择每个轴，使该轴向的方差在所剩方差中占的分额是最大的。

怎样来实现呢？给出一个合适的计算机程序并不难实现；给出合适的数学工具，也不难理解。从技术上来看，了解下面术语的读者计算数据点原始坐标的协方差矩阵（covariance matrix），并进行对角线化（diagonalize），找到特征向量（eigenvector）。这些就是转换后的空间上的轴，并按照特征值（eigenvalue）进行排序，因为每个特征值提供了这个轴向上的方差。

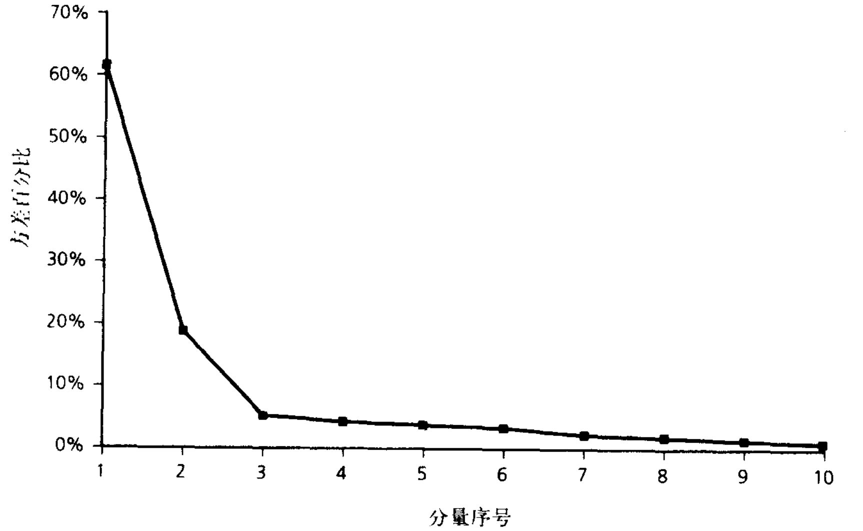
图7-5展示了一个含10个数值属性，即相应的数据点是在一个10维空间上的数据集的转换结果。想象一下原始数据点密布于一个10维的空间，这是我们画不出来的！选择方差最大的方向为第一个轴的轴向，第二个轴向选择与之正交且方差次大的方向，依次类推。图中列表按照被选择的次序，依次列出了沿每个新轴向上的方差。由于方差的总和是一个常量而与坐标系无关，因此用方差占总和百分比的形式列出。我们称轴为分量（component），每个分量要“担负”它在方差中的分额。图7-5b画出了每个分量所担负的方差对应于分量的序号。可以使用所有的分量作为新属性来进行数据挖掘，也可以只选择前面几个，即主分量（principal component），而丢弃其余的分量。在这个例子中，3个主分量担负了数据集84%的方差；7个便担负了95%以上。 307

对于数值型的数据集，在进行数据挖掘之前使用主分量分析，作为一种数据清理及属性生成的形式是很常见的。例如，你也许想要替代数值属性，用主分量轴或它们的一个子集来担负某个给定比例的方差，譬如说95%。注意属性的衡量尺度会影响主分量分析的结果，通常惯例是先将所有属性进行标准化，使之平均值为0且方差单元化。

另一种可能性是将主分量分析法递归地运用于决策树学习器中。普通的决策树学习器在每个阶段所选择的分裂都是沿平行于某个轴向的。然而，假设先进行了一次主分量转换，学习器选择了经过转换了的空间中的一个轴。这等同于沿原始空间中的某条斜线进行分裂。如果每次分裂之前都重新进行转换，结果将是一种多元决策树，它的分裂方向与轴或与其他分裂方向都是不平行的。

轴	方差	累计值
1	61.2%	61.2%
2	18.0%	79.2%
3	4.7%	83.9%
4	4.0%	87.9%
5	3.2%	91.1%
6	2.9%	94.0%
7	2.0%	96.0%
8	1.7%	97.7%
9	1.4%	99.1%
10	0.9%	100%

a) 每个分量的方差



b) 方差图

图7-5 数据集的主分量转换

7.3.2 随机投影

主分量分析将数据线性转换到低维空间。但代价昂贵。要找出转换（一个由协方差矩阵的特征向量所组成的矩阵）花费的时间将是维数的立方。对于属性数目庞大的数据集不可行。一个更为简便的方法是将数据随机投影到一个维数预先设定好的子空间。要找到随机投影矩阵是很容易的。但效果是否好呢？

事实上，理论表明随机投影能相当好地保存距离关系。这意味着它们可以和kD树或球树一同使用，在高维空间进行近似最近邻搜索。首先转换数据以减少属性数目，然后为转换了的空间建树。最近邻分类情形下，使用多个随机矩阵来建一个联合分类器能使结果更加稳定而且更少依赖于随机投影的选择。

为建标准分类器对数据进行预处理时，采用随机投影的效果不如经主分量分析仔细选择的效果，这并不出乎意外。但是，试验显示这些差别并不太大，而且随着维数的升高，差别呈减小趋势。当然，随机投影计算成本要低得多。

7.3.3 从文本到属性向量

在2.4节中曾介绍了包含文本的字符串属性并指出字符串属性值经常是一个完整的文件。字符串属性从本质来看是未指明属性值数目的名词性属性。如果只是简单地把它们当作名词性属性来处理的话，模型可依据两个字符串属性值是否相同来建立。但这种方式没有捕捉到任何字符串内在的结构，或者显示它所代表文本的任何有趣方面。 308~309

你可以想象将字符串属性中的文本分解为一个个段落、句子或词组。一般地，单词是最有用的单元。字符串属性中的文本通常是一连串的单词，文本所包含的单词通常可作为它最好的代表。例如，可将字符串属性转换为一系列的数值属性，每个单词用一个数值属性，代表这个单词出现的频率。这一系列单词，即一系列新属性是由数据集决定的，它的数量是相当大的。如果存在几个需要分别处理的字符串属性，新属性的名称必须区别开来，或许可采用自定义的前缀。

转变为单词（用tokenization）并不是表面看来那么简单的操作。记号可以由连续的字母排列形成、丢弃非字母字符。如果是数字，数字排列也要保留。数字可能牵涉到 + 号或 - 号、小数点以及幂次方，换句话说，就是它们要根据某种定义的数字语法排列。一个字母数字排列也许要被当作一个单独的记号。也许空格字符可作为记号的分隔符，也许white space（即包括tab 键和换行符）是分隔符，也许标点符号也是分隔符。句点符很难处理：有时它们应作为单词的一部分来考虑（与姓名首字母在一起、与称呼在一起、缩写以及数字），但有时又不能那样（如果它们是句子的分隔符）。连字号及省略号也有类似问题。

所有单词在被加入词汇表之前、也许都要先转变为小写。在预先设定的功能词或停止词（stopwords）、如the 、and 、but的清单上的词可以忽略。注意停止词清单是取决于语言的。事实上，大写习惯（德语大写的都是名词）、数字语法（欧洲使用逗号代表小数点）、标点符号习惯（西班牙语疑问句在句首加问号）、当然还有字符集本身都是取决于语言的。总之，文本是很复杂的。

低频率词譬如只用过一次的字句（hapax legomena[⊖]）经常被丢弃。有时在除去停止词之后，保留频率最高的k个单词，或者是为每个类别保留频率最高的k个单词，是有益处的。

有个问题伴随着所有这些tokenization 选项，即每个单词属性的属性值应是什么？属性值可以是单词累计数，这个单词在字符串中出现的次数、或者只是简单表明出现或未出现。可以对单词频率进行正常化使每个文件的属性向量具有相等的欧几里得长度。另一种方法是，将文件j中的单词i所出现的频率f_{ij}按照各种不同的标准方式进行转换。一种标准的对数词频率衡量便是$\log(1+f_{ij})$。在信息检索中广泛应用的衡量方法是 TF × IDF，即“词频率乘以文件频率倒数（term frequency times inverse document frequency）”。这里，词频率被一个因数调整，这个因数取决于这个词被其他文件运用到的普及度。TF × IDF尺度的标准定义如下 310

$$f_{ij}\log\frac{\text{文件数量}}{\text{含有词}i\text{的文件数量}}$$

主要想法是文件的特性基本上是由其中经常出现的单词而定，在公式中占据第一个因子；除去那些在每个文件或几乎每个文件中都用到的、对于鉴别毫无用处的单词，这占据了公式的

⊖ A hapax legomena 是指在全文中只出现过一次的单词。

第二个因子。TF × IDF不仅仅特指这个公式，而且泛指这一种类的衡量方法。例如，频率因子f_{ij}也可以用对数词频率$\log(1+f_{ij})$来代替。

7.3.4 时间序列

在时间序列数据中，每个实例代表不同的时间间隙，属性给出了与该时间间隙所对应的值，如气象预报或股市行情预测。有时需要能将当前实例的一个属性值用过去的或将来的实例所对应的属性值来替换。更常用的是用当前实例与过去实例属性值的差值来替换当前的属性值。例如，当前实例与前一个实例属性值的差值（差值常被称为Delta）通常比属性值本身含有更多信息量。第一个实例由于时间位移值未知，可以删除或用残缺值来代替。差值从本质上来说是以
311 由时间间隙大小所决定的某个常量为量度的第一次求导，连续的Delta转换即为更高次的求导。

在一些时间序列中，实例不代表定期的样本，而每个实例的时间是由时间戳（timestamp）属性给出的。不同的时间戳之间的差别在于实例的时间间隙大小不同，如果要取其他属性的连续差值，必须除以间隙大小以使求导正常化。另一种情形是每个属性可以代表不同的时间，而非每个实例代表不同时间，因此时间序列是从一个属性到下一个属性，而非从一个实例到下一个实例。那么，如果需要差值，必须取每个实例的一个属性和下一个属性之间的差值。

7.4 自动数据清理

一个令实际数据挖掘工作头疼的问题便是数据的质量低劣。在大型数据库中错误更是常见。属性值、类值经常是不可靠和错误的。一种解决此问题的方法是艰辛地检查数据，然而数据挖掘技术本身也能对此问题的解决有所帮助。

7.4.1 改进决策树

一个令人惊讶的事实是利用训练数据进行决策树的归纳，可以简单化且不损失正确率，通过丢弃被错误分类的训练实例，重新学习，然后重复直到没有错误分类的实例为止。在一些标准数据集上的试验显明，这几乎不影响标准的决策树归纳法C4.5的分类正确率。有时性能略有提高，有时略微变差。差别不显著，即使有显著差别，两者都有优势可能。这种技术影响的是决策树的大小。虽然性能表现大致相当，最终的决策树比原来的总是小得多。

这是什么原因呢？当决策树归纳要修剪掉一个子树时，它应用统计测试通过数据来判定这个子树是否“合理”。修剪决定相信在训练实例分类的正确性上做出少量牺牲，将提高在测试集上的性能表现。一些未修剪树能正确分类的训练实例，现在用经过修剪的树将会被错误分类，实际上，决策树将忽略这些训练实例。

然而，这个决策只是应用于局部，只在被修剪的子树中。它的影响没有被允许往树的上层渗透，那样也许会造成选择出不同的分支属性。从训练集中去除被错误分类的实例并重新学习决策树，使修剪决策做出合理的结论。如果修剪策略较好，不会破坏性能，甚至由于允许选择更好的属性还能提高性能。

毫无疑问，进行专家咨询效果更好。对错误分类的训练实例进行验证，发现实例是错误的可将其删除，或者还可使用，则进行修正。

注意我们假设实例没有出现任何系统上的错误分类。如果实例在训练集和测试集上都被系统性地破坏，例如，一个类值可能被另一个类值替换了，只能期望在错误的训练集上训练
312 会在（同样也是错误的）测试集上产生较好的性能。

有趣的是人们发现当往数据中人为地添加属性干扰时（不是类干扰），如果在训练集中也添加了同样的属性干扰，在测试集上的性能会提高。换句话说便是当存在属性干扰问题时，如果性能测试将要在“不清洁”的数据上进行，那么用一个“清洁”的训练集来训练并不好。有一种学习方法能够学习对属性干扰进行稍许补偿。从本质上来说，它能学习哪些属性不可靠，如果都不可靠，怎样最好地利用它们产生一个更可靠的结果。将训练集上的属性干扰去除便失去了怎样最好地抗干扰的学习机会。但是对于类干扰（而非属性干扰），如果可能，最好还是使用无干扰的实例进行训练。

7.4.2 稳健回归

历年来人们已经知道了干扰数据在线性回归中所造成的问题。统计学家们经常是靠检查数据中的孤立点（outlier）并人工将它们去除。在线性回归情形中，孤立点可从视觉上辨别出来，虽然不是完全清楚这个孤立点是一个错误，或者只是一个不寻常、却是正确的值。孤立点大大影响了最小二乘回归（least-squares regression），因为二乘方的距离衡量加强了远离回归线的数据点的影响。

处理孤立点的统计方法称为稳健型（robust）。使回归更为稳健的一种方法是采用绝对值距离衡量来代替通常使用的二乘方距离衡量。这削弱了孤立点的影响。另一种可能的方法是试图自动识别孤立点，把它驱逐出考虑范围。例如，可以形成一条回归线，然后驱逐离回归线较远的10%的数据点。第三个方法是使距离回归线二乘中值（median）（而非平均值）最小化。结论是这种估计器非常稳健，真正是既在X 轴向对孤立点进行处理，又在孤立点常规考虑方向Y轴向进行处理。

用于描述稳健回归的常用数据集是从1950年到1973年比利时的国际长途电话图，如图7-6所示。这个数据集来源于比利时经济部发布的比利时统计调查。这个图似乎显示历年来（国际长途电话数量）呈上升趋势，但从1964年到1969年这段时间数据点反常。实际上是这段时间的数据被错误地记录为电话总分钟数。1963年和1970年也受到部分影响。这个错误造成孤立点在Y轴方向上的巨大偏差。

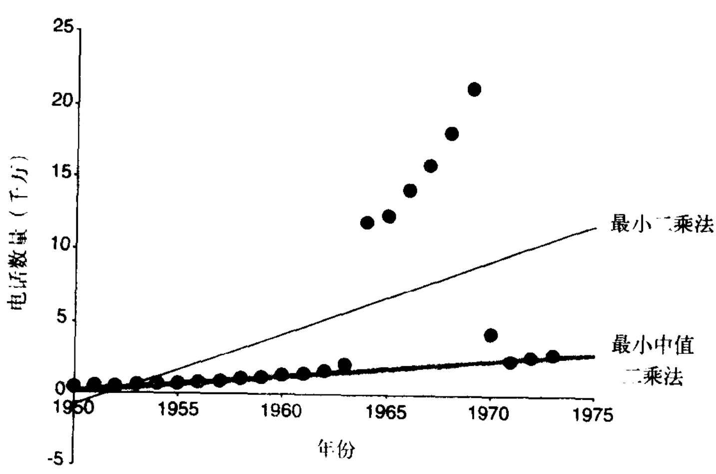


图7-6 比利时国际长途电话数量，1950~1973年

由于这些反常数据，最小二乘回归线受到严重影响，这并不奇怪。然而，最小中值二乘回归线却明显不受干扰。对于这条线有一个简单而自然的说明。从几何学角度看，它相当于

寻找一条覆盖半数观察点的最窄带，带的厚度是从垂直方向衡量的。这条窄带在图7-6中显示为灰色，需要仔细看。最小中值二乘回归线则位于这条窄带的中央。这个术语比常规的最小二乘回归定义更容易解释、显示。不幸的是，基于中值的回归技术有个严重缺陷：它们造成很高的计算成本，面对实际问题经常是不可行的。

7.4.3 侦察异情

任何自动侦察明显错误数据的一个严重问题是把有价值的东西和不需要的东西一起扔掉。由于缺乏咨询专家，没有办法知道某个实例真的是一个错误或者只是所应用的模型不适合它。在统计回归中，使之可视化能有所帮助。如果要拟合的是一条错误的曲线，即使不是专家通
313~314 常也能明显看出，比如要使一条直线拟合位于抛物线上的数据。图7-6的孤立点当然是非常明显，但大多数的问题并不那么明显，“模型种类”比回归线要敏感。虽然对于大多数的标准数据集来说，丢弃不符合决策树的实例能获得较好的结果，但在处理某个新数据集时，这不一定是很合适的。也许新数据集只是不适合用决策树模型。

一种人们已经尝试的方法便是使用几种不同的学习方案，如用一个决策树、一个最近邻学习器和一个线性判别函数（linear discriminant function）来过滤数据。保守的策略是用这三种方法分类都失败时方可判定一个实例是错误的并将其从数据中去除。有时用这种方法过滤数据，然后使用经过滤的数据作为最终学习方案的输入，这与简单地使这三个学习方案，然后进行投票产生最终结果相比较，能获得更好的性能。三种学习方法在经过滤的数据上进行训练，然后投票，这样能产生更好的效果。然而，投票技术存在一个危险，某些学习算法比较适合某种类型的数据，因此最适合的方法也许能决定投票结果！我们将在下一节考察一种更为精细的方法来组合不同的分类器的输出结果，称为堆栈法。如平时一样，了解数据，并用不同的方法来考察数据。

过滤方法的一个可能危险是它们可能会牺牲某个类别（或一组类别）的实例来提高剩余类别的正确性。没有什么通用的方法来防止这一点，实践发现这是个常见问题。

最后，值得再次提醒注意的是首先要尽力得到合适数据，自动过滤只是一种作用极其有限的替代。如果在实践中太耗时间和成本，可以用人工检查那些由过滤器鉴别出来的可疑实例。

7.5 组合多种模型

明智的人们在做出某个关键决策时，通常是要考虑一些专家们的意见而不是只依赖于自己的判断或依赖于某个孤立的、被信任的顾问。例如，在选择一个重要的新政策方向之前，一个好的独裁者会广泛地征询意见，盲目地听从一个专家的意见是不明智的。在民主的条件下，对不同的观点进行讨论也许可以达成一致意见，如果不能还可以进行投票。不管采用哪种方法，不同专家意见被组合在一起。

在数据挖掘中，由机器学习产生出来的一个模型可以被看作是一个专家。用专家这个词也许过于强化了！这取决于训练数据的数量和质量以及学习算法是否适合于手头的问题，这
315 个专家也许很令人遗憾、很无知，但不管怎样，我们还是先用这个术语。很明显，能使所做出的决策更为可靠的方法便是将不同的输出模型组合起来。一些机器学习技术是通过学习合成模型将它们组合应用来实现，其中最主要的方法有装袋、提升（boosting）和堆栈。多数情形下，与单个模型相比较，它们都能使预测性能有所提高，并且是可用于数值预测以及分类

预测的通用技术。

装袋、提升和堆栈是过去十年发展起来的，它们的性能表现常常出人意料的好。机器学习研究者曾努力分析其中的原因。在这个努力过程中，又有新方法出现，有时能带来更好的结果。例如，人类的委员会（制度）很少能从干扰中获益，与此相反，添加随机的分类器变体重新组合后的装袋法却能获得性能的提高。更进一步的研究分析揭示出提升法，也许是这三者中最好的方法，是与统计技术的迭加模型非常相关的，对这个方法的认识也使程序得到了改善。

这些组合模型都难以分析这个弊端：它们可以由许多甚至是几百个单个模型组成，虽然性能表现不错，但这些决策的改善归功于哪些因素并不易了解到。近几年发展出一些方法将委员会的益处和易理解的模型融和在一起。有些产生标准的决策树模型，另一些产生能提供可选路径的决策树新变体。

作为结束，我们引入另一种技术，使用误差纠正输出编码组合模型的技术。与其他三种技术相比较，这个方法更加专业化，只适用于分类问题，并且是含三个以上类别的问题。

7.5.1 装袋

组合不同模型的决策意味着由不同的输出结果合并出一个预测。对于分类问题，最简单的方法就是进行投票（也许是带有权值的投票）；对于数值预测问题，就是计算平均值（也许是带有权值的平均值）。装袋法和提升法都采纳这种方式，但它们用不同方法得到各自的模型。在装袋法中模型都有相同的权值，而在提升法中，要给较成功的模型加权，就像执行主管对于不同专家的建议依据它们的经验程度给予排位值一样。

为了介绍装袋法，假设从问题领域中随机选择几个相同大小的训练数据集。想象使用某种特定的机器学习技术来为每个数据集建决策树。你也许期望这些树在实践中是一样的，对于每个新的实例会做出同样的预测。令人惊讶的是，这个设想是相当错误的，特别是当训练数 316
据集相当小时。这是个令人烦扰的事实，在整个计划上投下了阴影！原因是决策树归纳法（至少，第4章中所描述的标准的从上而下的方法）是一个不稳定的过程，训练数据的稍许变化易导致在某个节点处不同的属性被选择，使这个节点下部的分支结构出现明显的差异。这意味着对于一些测试实例，部分决策树能产生正确的预测，部分却不能。

回到前面的专家比喻，将每个专家想象为单个的决策树。我们可以通过让它们对每个测试实例投票来组合这些树。如果一个类收到了比其他类更多的投票，它就被当作是正确的类别。一般地，投票数越多越好，考虑越多的投票，由投票所决定的预测便越是可靠。如果有新的训练数据被发现，用它们建树并让预测结果参与投票，决策结果很少会变差。特别是组合分类器的正确率很少比由单个数据集建立的决策树所表现的正确率差。（然而改善并不是确保的，从理论上可以看到，组合决策变得更差也是有可能存在的情形。）

组合多种假设的效果可以用一种称为偏差-方差分解（bias-variance decomposition）的理论来考察。假设我们有无数个相同大小的独立训练集数据，用它们来生成无数个分类器。用所有的分类器对一个测试实例进行处理，由多数投票来决定答案。在这个理想状况下，还是会出现错误因为没有一种学习方案是完美的，误差率是由机器学习方法与手头问题的适配程度所决定的，而且总是存在干扰数据的影响，这也不可能学习到。假设预期的误差率是用组合分类器在无数个独立测试实例上的平均误差评估出来的。某个具体学习算法的误差率称为

学习算法对于这个学习问题的偏差，是学习方法与问题适配程度的一种衡量。这个技术定义是对1.5节中所介绍的偏差这个模糊的术语进行量化的一种方法，它衡量了一种学习算法的“永久性”误差，这个误差即使考虑无数个训练集也是无法消除的。当然在实践运用中不可能精确计算，只能大致估算。

在实践中，学习模型的第二个误差来源是所使用的具体的训练集，它是有限的，因此不能完全代表真实的实例集。这个误差部分的期望值来源于所有给定大小的可能训练集以及所
317 有可能测试集，称为学习方法对于这个问题的方差。一个分类器的总期望误差是由偏差和方差这两部分之和所构成：这便是偏差-方差分解（bias-variance decomposition）[⊖]。组合多种分类器能通过减小方差项从而减小期望误差值。越多的分类器参与进来，方差减小量也越大。

当要将这种投票法运用于实践中时，困难出现了：通常只有一个训练数据集，要获得更多的数据不是不可能就是代价太大了。

装袋法试图使用一个给定的训练集模拟上述过程，来缓解学习方法的不稳定性。删除部分实例并复制其他的实例来改变原始训练数据，而不是每次采样一个新的、独立的训练数据集。从原始数据集中随机采样实例、产生出一个新的同样大小的数据集，这个采样程序不可避免地出现一些重复实例，删除另一些实例。如果觉得这个想法似曾相识，那是因为在第5章描述自引导方法用于估计一个学习方法的推广误差时（5.4节）曾经介绍过。事实上，术语装袋代表自引导整合（bootstrap aggregating ）。装袋法将学习方案，例如决策树，应用于每一个人工生成的数据集上，从中形成分类器并对预测类进行投票。图7-7对这个算法进行了概述。

装袋和先前所述理想化的程序之间的差别在于训练数据集形成的方法不同。代替从领域中获得独立的数据集，装袋只是对原始数据集进行重新取样。重新取样的数据集当然是各不相同，但肯定不是独立的，因为它们都是基于一个数据集的。然而，结果是装袋所产生的组合模型的性能经常比在原始训练数据集上产生的单个模型有明显的改善，而且没有明显变差的情形出现。

```
模型生成
令n为训练数据的实例数量
对于t次循环中的每一次
    从训练数据中采样n个实例
    将学习算法应用于所采样本
    保存结果模型
分类
对于t个模型中的每一个
    使用模型对实例进行类预测
返回被预测次数最多的类
```

图7-7　装袋算法

装袋法也可用于进行数值预测的学习方法。例如，模型树。差别只是将各个预测（都是实数）进行平均计算、来代替对结果进行投票。偏差-方差分解也适用于数值预测，分解对于新数据所做预测的均方误差的期望值。偏差定义为对所有模型进行平均时的期望均方误差，

⊖　这是一种简化的方法。可以从其他文献中找到另外几种不同的方法用于执行偏差-方差分解，对此还没有统一的方法。

这些模型是在所有可能的、相同大小的训练数据集上所建的；方差是单个模型的期望误差，这个模型是在某个具体的训练数据集上所建立的。从理论上可以看出，平均多个建立在独立训练集上的模型总是可以减小均方误差期望值。（正如我们先前提到的，这种模拟结果对于分类来说不是完全真实的。）

7.5.2 考虑成本的装袋

当学习方法不稳定时，即输入数据的小变化能导致生成差别相当大的分类器，装袋可以为此提供最大的帮助。实际上，尽可能地使学习方法不稳定，有助于增加合成分类器中的差异性（diversity）。例如，对决策树使用装袋技术，决策树是不稳定的，如果不对树进行修剪使决策树更不稳定时经常会获得更好的性能。另一种改善可通过改变分类预测组合的方法来获得。装袋原本是使用投票法，但如果模型可以输出概率估计而不只是简单的分类，那么凭直觉将这些概率取平均值来替代投票是有意义的。这样做不仅经常能稍许改善分类性能，而且能使装袋的分类器产生一个概率估计，经常比单个模型所产生的更加精确。因此装袋的实现通常采用这种方法来组合预测。

在5.7节我们展示了怎样通过最小化预测的期望成本使一个分类器对成本敏感。正确的概率估计是必要的，因为要用它们来获取每个预测的期望成本。装袋是成本敏感分类的最佳候选，因为它能从决策树和其他一些功能强大但不稳定的分类器中产生非常精确的概率估计。但是，缺点是应用了装袋 技术的分类器很难分析。

一种称为MetaCost的方法将装袋的预测益处和一个易理解的模型组合起来，适用于成本敏感的预测。它采用装袋技术建立一个合成分类器，用这个分类器对训练数据重新赋予标签，它根据装袋所获得的概率估计，赋予每个训练实例一个类预测使期望成本最小化。MetaCost 318~319
随后丢弃原先的类标签，从重新标签的数据中学习出一个新的模型，比如一个修剪的决策树。新模型会自动考虑成本，因为已经含在类标签中！结果是一个成本敏感的分类器可用于分析预测是如何获得的。

除了刚刚提到的成本敏感分类技术，在5.7节中还描述了一种成本敏感学习方法，它是通过改变每个类在训练数据中的比例反映到成本矩阵上，来学习一个成本敏感分类器。MetaCost 似乎比这种方法能获得更精确的结果，但它需要更多的计算。如果不需要得到一个易于理解的模型，MetaCost 的后加工程序便是多余的，最好直接使用经装袋后的分类器与最小预期成本法共同协作。

7.5.3 随机化

装袋法将随机概念引入学习算法的输入中，产生一个不同的合成分类器，经常带来极好的效果。然而，还有其他方式应用随机化来产生差异性。有些学习算法本身已带有一个内置的随机机构。例如，使用反向传播算法学习一个多层感知器时（如6.3节所述），网络权值被设定为一个随机选择的小数值。所学的分类器依赖于这个随机数值，因为（根据这个值）算法会找到一个不同的误差函数的局部最小值。要使分类结果更加稳定的一种方法是使用不同的随机数多次重复学习过程，然后将多个分类器的预测结果通过投票或平均的方法组合起来。

几乎所有的学习方法都含有某种意义上的随机化。考虑一个算法要在每步贪心式地挑选最好的选择，如决策树学习器在每个节点处要挑选最好的属性来进行分裂。可以通过在N个最好的选择中随机挑选一个，来替代原先只能有单个胜者的策略，或者随机选择一个属性子集

然后从中挑选出最好的属性，从而将算法实现随机化。当然，这里存在一个权衡问题：随意性越大产生的学习器越具有差异性，而另一方面却更少利用数据（信息），可能会造成单个模型的正确率下降。最好的随机范围只有通过试验才能限定。

虽然装袋和随机化技术产生相似的结果，但要将它们组合起来需要付出一点代价，因为它们引入随机的方法不同，或许是互补型的。一个用于学习随机森林（random forests）的流
320 行算法在装袋的每一轮循环中建立随机化的决策树，通常能带来出色的预测结果。

由于必须对学习算法进行修改，随机化比装袋法需要更多的工作量，但是这种技术能适用于更多种不同类型的学习器。前面我们已经注意到，装袋法对于那些输入发生小变化而输出对此不敏感的稳定学习算法是没有用处的。例如，在最近邻分类器上应用装袋技术便是毫无意义的，即使采用重新取样将训练数据打乱，它们的输出几乎没有什么改变。然而，随机化却仍然适用于稳定的学习器，秘诀在所选用的随机化方式能在不牺牲太多性能指标的前提下使分类器具有差异性。最近邻分类器的预测取决于实例之间的距离，严重依赖于选择哪些属性来计算距离，因此最近邻分类器可通过使用不同的、随机挑选的属性子集来实现随机化。

7.5.4 提升

装袋法充分利用了学习算法内在的不稳定性。从直觉上看，只有当各个模型之间存在明显差异，并且每种模型都能正确处理一定合理比例的数据时，组合多种模型才能发挥用处。理想状况是这些模型对其他模型来说是一个补充，每个模型是这个领域中某一部分的专家，而其他模型在这部分却不能很好表现，就像是执行官要寻觅那些技能和经验互补的顾问，而不是互相重复的。

用于组合多种模型的提升方法通过明确地搜寻与另一个模型互补的模型来挖掘这个内在的见解。首先，相似点是与装袋一样，提升利用投票（用于分类）或者取平均值（用于数值预测）来组合各个模型的输出。另外，也同装袋一样，它组合同一类型的模型，例如决策树。然而，提升是循环迭代的。在装袋中各个模型是单独建立的，而提升的每个新模型是受先前已建模型的性能表现影响的。提升法鼓励新模型成为处理先前模型所不能正确分类的实例的专家。最后一个不同点是提升根据模型的性能来对每个模型的贡献加权，而不是赋予每个模型同等的权值。

提升存在许多不同的实现方法。这里描述一种广泛应用的、专门用于分类的、称为AdaBoost.M1的方法。像装袋技术一样，它适用于任何的分类学习算法。为使问题简单化，假设学习算法可以处理带有权值的实例，实例的权值为正数。(在后面我们还会来看这个假设。)实例权值的出现改变了分类误差的计算方法，误差等于错误分类实例的权值总和除以所有实例权值总和，替代了原来错误分类实例的比例。通过实例加权，学习算法可以将精力集中于
321 特定的实例集上，即那些权值较高的实例。这些实例特别重要，因为存在较大的动力要对它们正确分类。6.1节中讲述的C4.5算法是无需修改便能适合加权实例的学习方法的一个样例，因为它已使用了分数实例（fractional instances）观念来处理残缺值。

图7-8总结了提升算法，首先赋予所有训练实例相同的权值，然后应用学习算法在这个数据集上生成一个分类器，再根据这个分类器的分类结果对每个实例重新加权。减小正确分类的实例权值，增加错误分类的实例权值。这产生了一组权值较低的“容易对付”实例和另一组权值较高的“难以对付”实例。在下一轮循环以及所有以后的循环中，分类器都是建立在

经重新加权的数据上，并且专注于对那些难以对付的实例进行正确分类。然后依据这个新分类器的分类结果增加或减小实例的权值。结果是部分较难对付的实例可能变得更难，部分较易对付的实例可能变得更易；另一方面，其他较难对付的实例可能变得较易，较易对付的实例可能变得较难对付了，在实践中各种可能都会发生。在每一轮循环后，权值反映出目前所生成的分类器对实例的错误分类有多频繁。通过对每个实例“难度”的衡量，这个程序提供了一种巧妙的方法来生成一系列互补型的专家。

```
模型生成
赋予每个训练实例相同的权值。
t次循环中的每一次：
    将学习算法应用于加了权的数据集上并保存结果模型。
    计算模型在加了权的数据集上的误差e并保存这个误差。
    如果e等于0或者e大于等于0.5：
    终止建模型。
    对于数据集中的每个实例：
        如果模型将实例正确分类：
            将实例的权值乘以e / (1 - e)。
    将所有的实例权值进行正常化。
分类
赋予所有类权值为0。
对于t（或小于t）个模型中的每一个：
    给模型所预测的类加权 - log (e / (1 - e))。
返回权值最高的类。
```

图7-8 提升算法 322

每轮循环后，权值应做多大的调整呢？这个答案依赖于当前分类器的总体误差。明确地说，如果e代表分类器在加权数据上的分类误差（在0和1 之间的一个小数），那么对于正确分类的实例，权值更新为

$$\text{权值} \leftarrow \text{权值} \times e / (1-e)$$

对于错误分类的实例，权值保持不变。当然，这并没有如前所述，增加被错误分类的实例的权值。然而，在更新了所有实例的权值后，要进行正常化处理，使它们的权值总和与原来的相同。每个实例的权值都要除以新权值总和再乘以原来的权值总和。这样便自动增加了每个错误分类实例的权值，同时减小了每个正确分类实例的权值。

每当在加权的训练数据集上的误差率大于等于0.5时，提升程序将删除当前的分类器并不再继续进行循环。当误差率等于0时，也同样处理，因为这时所有实例的权值都为0。

我们已经介绍了提升方法是如何生成一系列分类器的。为了做出一个预测，使用一个加了权的投票来组合它们的输出。要决定这些权值，注意那些在加了权的训练数据上，用于建立该分类器的数据，性能表现好的分类器（e接近于0）应当获得一个高的权值，而性能表现差的分类器（e接近于0.5）则应获得一个较低的权值。更具体地说

$$\text{权值} = -\log \frac{e}{1-e}$$

这是一个在0和无穷大之间的正数。顺便提一下，这个公式也解释了为何在训练数据上性能表现完美的分类器要删除，因为当e为0时，权值无定义。为了做出预测，将投给某个具体类的

所有分类器的权值相加，选择相加总和最大的那个类别。

从开始我们便假设学习算法能够处理加权实例。在6.5节最后的局部加权线性回归部分中已解释了如何调整学习算法以便处理加权实例。也可以通过重新采样，即如同装袋法中使用的技术，从加权的数据集中产生一个不考虑权值的数据集，从而不用调整学习算法。在装袋法中，每个实例被选择的概率是等同的；而在提升法中，实例被选择的概率按它们各自的权值比例。结果是权值高的实例重复的频率高，而权值低的实例可能都不会被选择。一旦新数据集与原始数据集达到同样大小，便将新数据集传给学习方法，而不使用加权的数据集。正
323 是如此简单。

这个程序的一个缺陷是有些权值低的实例不会被选入重新采样的数据集中，因此造成在应用学习算法前，部分信息已丢失。然而，这也可以成为一种优点。当学习方法生成了一个误差大于0.5的分类器时，如果直接使用加权的实例，提升必须终止；而如果是采用重新取样的方法，可以丢弃目前重新取样的数据集，使用不同的随机种子再次重新取样，生成一个新的数据集，那么便有可能会生成一个误差低于0.5的分类器。有时，使用重新取样的方法比原先采用加权方法的算法要进行更多次的提升循环。

提升技术的想法起源于机器学习研究的计算学习理论分支。理论研究者对提升感兴趣是因为它使获得性能保证成为可能。例如，可以看到随着循环次数的增加，在训练数据上的组合分类器误差很快地向0靠拢（与循环次数呈指数级的速度加快）。不幸的是，正如5.1节中所述，在训练集上的误差保证并不十分令人感兴趣，因为这并不表示在新数据上会有好的性能表现。但是从理论上看，提升技术只是当各个分类器对于所呈现的总体训练数据来说太过“复杂”，或者训练误差变得太大、太快时，才会在新数据上失败（Schapire等人1997年对此做了精确解释）。照例，问题在于要找到各个模型的复杂度和它们与数据的拟合度之间的恰当平衡点。

如果提升法在新的测试数据上能成功地减小误差，它常常是以一种场面浩大的方式。一个非常令人惊讶的发现是当在训练数据上的组合分类器误差降到0后，持续进行更多次的提升循环，能够在新数据上减小误差。研究者对这个结论感到吃惊，因为它似乎与Occam剃刀原理相矛盾，Occam剃刀原理宣称在两个能同样好地解释试验证据的假设中，简单的那个优先。进行更多次的提升循环，不减少训练误差并没有对训练数据做出任何更好的解释，但肯定增加了分类器的复杂度。所幸的是，考虑分类器在所做预测上的置信度可以解决这个矛盾。这个置信度是根据真实类的估计概率和除了真实类、最有可能的预测类的估计概率之间的差别来衡量的，称为边差（margin）的量。这个边差越大，分类器能预测出真实类的置信度便越大。结论是提升在训练误差降到0之后能增大这个边差。画出达到不同提升循环数时，所有的训练实例的累积边差分布，可以看到这个效果，这个图称为边差曲线（margin curve）。因此，
324 如果考虑边差量来解释试验证据，Occam剃刀原理还是同样的锋利。

提升法的一个好处是能在重新加权数据上误差低于0.5、非常简单的分类器中产生出一个功能强大的组合分类器。通常，这个方法非常简单，特别是对于二类学习问题！这些简单的学习方法称为弱（weak）学习器，提升法将弱学习器转变为强学习器。例如对于二类问题，将提升应用于只有一层的、非常简单的决策树称为决策树桩（decision stump），可以获得好结果。另一个可能方法是将提升应用于学习单个联合规则的算法，譬如决策树上的一条路径，实例的分类是依据这条规则是否覆盖了这些实例。当然，多类数据集误差率要达到低于0.5 更

为困难些。提升技术也可以应用在决策树上，但通常比决策树桩更加复杂。一些更为复杂的算法也已发展起来，它们能对非常简单的模型应用提升技术，在多类情形中获得成功。

与装袋技术相比较，应用提升技术经常能产生出在新数据上明显精确的分类器。但是不像装袋，提升有时在实际情形中会失败，它会产生出一个分类器，性能明显差于在同样数据上建立的单个分类器。这表明组合分类器和数据过度拟合了。

7.5.5 叠加回归

提升研究一开始便激起了研究者们强烈的兴趣，因为它能从表现一般的分类器中获得一流的性能。统计学家很快发现它可被重建成一个叠加模型的贪心算法。叠加模型在统计学领域有了相当长的历史。一般来说，泛指任何将源于其他模型的贡献相加起来产生预测的方法。大多数用于叠加模型的算法不是独立建立基本模型的，而是要保证与另一个模型互补，并且要根据某些特定标准来形成优化预测性能的合成模型。

提升是执行正向分段叠加建模（forward stagewise additive modeling）。这类算法由一个空的合成模型开始，相继合并新成员。在每个阶段，加入能使总合成模型预测性能达到最好的模型，而不改变已经包括在合成模型中的成员。对合成模型的性能优化意味着下一个模型要专注于那些在合成模型上表现较差的训练实例。这正是提升所做的，即赋予这些实例更大的权值。

这里介绍一个众所周知的、用于数值预测的正向分段叠加建模方法。先建立一个标准的回归模型，如一个回归树。在训练数据上的误差即在预测值和观测值之间的差别，称为残差（residuals）。然后学习第二个模型来纠正这些误差。也许用另一个回归树预测观测残差。为了达到这个目的，在学习第二个模型之前用它们的残差来替代原始的类值。将第二个模型所做 325
出的预测叠加到第一个预测上便自动降低了在训练数据上的误差。通常某些残差仍然存在，因为第二个模型也不是完美的，因此继续建立第三个模型来学习预测残差的残差，依此类推。这个程序令人回想起在3.5节中所见的用于分类的、带有例外的规则。

如果单个模型能使预测的平方误差达到最小值，正如线性回归模型那样，那么这个算法总体上能使整个合成分类器的平方误差达到最小值。在实践中当基本学习器使用一种启发式近似的方法效果也不错，譬如用6.5节中介绍的回归和模型树学习器。实际上，在叠加回归中使用标准的线性回归作为基本学习器是毫无意义的，因为线性回归模型的总和还是一个线性回归模型，并且回归算法本身便能使平方误差最小化。然而，如果基本学习器是一个基于单个属性的、使平方误差最小化的回归模型，情况就不同了。对照称为多重线性回归的标准多属性方法，在统计学上称它为简单线性回归。事实上，联合使用叠加回归和简单线性回归，并且重复迭代直到合成分类器的平方误差不再降低，便产生一个叠加模型，与最小平方多重线性回归函数是相同的。

正向分段叠加回归法有过度拟合的倾向，因为叠加的每个模型越来越与训练数据拟合。使用交叉验证法来决定何时停止。例如，对每种不同循环次数进行交叉验证，次数达到用户指定的某个最大值，选择其中能使平方误差的交叉验证估计达到最小值的那个。这是个好的停止标准，因为交叉验证产生了一个对未来数据相当可靠的误差估计。另外，使用上述方法并联合简单线性回归作为基本学习器，能有效地将多重线性回归和内置的属性选择组合起来，因为如果能降低交叉验证误差，它只会包含下一个最为重要的属性贡献。

为了实现方便，正向分段叠加回归法通常从简单预测训练数据平均类的0层模型开始，从而使每个后继模型与残差拟合。这也暗示了防止过度拟合的另一种可能：替代原先扣除模型的整个预测值以产生下一个模型的目标值，在扣除之前将预测值乘以某个用户指定的在0和1之间的常量来减小预测值。这降低了模型对残值的拟合，从而减少过度拟合的机会。当然，这也可能增加要获得一个好的叠加模型所需的循环次数。减小乘数能有效地衰减学习过程，增加在恰当时刻停止的机会，但也增加了运行时间。

7.5.6 叠加logistic 回归

叠加回归也如同线性回归一样可应用于分类问题。然而从4.6节中了解到对于分类问题logistic 回归优于线性回归。通过修改正向分段建模方法来对叠加模型进行类似的调整，便可进行叠加logistic回归（additive logistic regression）。如同在4.6节中那样，利用对数转换将概
326 率估计问题转换为一个回归问题，并像对待叠加回归那样使用一个合成模型（如回归树）来完成回归任务。在每个阶段，添加对于某个给定的合成分类器能使数据的概率达到最大值的模型。

假设f_j是合成分类器中的第j个回归模型，$f_j(\mathbf{a})$是这个模型对实例**a**的预测。假设有一个二类问题，要使用叠加模型$\Sigma f_j(\mathbf{a})$来获得第一个类别的概率估计：

$$p(1|\mathbf{a})=\frac{1}{1+e^{-\Sigma f_j(\mathbf{a})}}$$

这和4.6节所用的表达式非常类似，这里使用了缩写，用向量来表示实例**a**，并且原来的加权属性值总和被替换成为任意复杂的回归模型f的总和。

图7-9展示了二类问题的LogitBoost 算法，它进行叠加logistic 回归并产生个体模型f_j。这里对于属于第一个类别的实例，y_i为1；对于属于第二类别的实例，y_i为0。在每轮循环中，这个算法要使回归模型f_j拟合原始数据集的加权版本，这个加权版本基于假设的类值z_i和权值w_i。我们假设$p(1|\mathbf{a})$是根据前一轮循环所建的f_j计算出来的。

模型生成
```
对于j=1到t次循环:
    对于每个实例a[i]:
      将回归的目标值设定为
          z[i] = (y[i] - p(1|a[i])) / [p(1|a[i]) × (1 - p(1|a[i]))]
      将实例a[i]的权值设定为
          p(1|a[i]) × (1 - p(1|a[i]))
    使回归模型f[j]与类值为z[i]、权值为w[i]的数据相拟合。
```
分类
```
如果p(1|a) > 0.5，预测结果为第一类别，否则为第二类别。
```

327 图7-9　叠加logistic回归算法

这个算法的推导过程在此书的讨论范围之外，但是可以看出如果每个模型f_j是由相应的回归问题的最小平方误差所决定的，这个算法是依据合成模型使数据概率最大化。实际上，如果用多重线性回归来形成f_j，算法会在最大似然线性logistic回归模型处收敛，它是4.6节中所述的迭代重新加权的最小平方方法的化身。

从表面上看，LogitBoost 和AdaBoost差别相当大，但是它们所形成的预测器主要差别是

在前者直接优化似然，而后者优化一个指数级的损失函数，这个函数可以看成是似然的近似值。从实践角度来看，差别是LogitBoost使用了回归方法作为基本学习器，而AdaBoost 与分类算法一起使用。

我们只讲述了用于二类问题的LogitBoost方法，然而这个算法还可以推广到解决多类问题。如同叠加回归一样，可以用一个预先设定的乘数来减小单个模型f_j 的预测值，并且利用交叉验证来决定一个适当的循环数目，从而降低过度拟合的危险。

7.5.7 选择树

装袋、提升以及随机化都生成合成分类器。很难分析何种信息从数据中被提炼出来。如能生成具有相同预测性能的单个模型将是很合人意的。一种可能是生成一个人工数据集，它是通过从实例空间随机采样数据点并根据合成分类器的预测来赋予它们类表签，然后从这个新数据集上学习出一个决策树或规则集。为了能从决策树中获得与合成分类器相似的预测性能，可能需要一个大型的数据集，但是起码这个策略要能够复制合成分类器的性能，假如合成分类器本身包含决策树，那么它肯定能做到。

另一种方法是取得能简洁地代表一个合成分类器的单个结构。如果合成分类器是由决策树组成，这便能实现，它的结果称为选择树（option tree)。选择树与决策树的不同之处在于它们包含两种类型的节点，决策节点和选择节点（option nodes)。图7-10展示了天气数据的一个简单例子，它只带有一个选择节点。要对一个实例进行分类，将其随树向下过滤。在决策节点处通常只选择一个分支，但是在选择节点处则选择所有的分支。这意味着实例最终以一个以上的叶节点而告终，必须从这些叶节点所获的分类结果中组合出一个总体分类结果。这 328
可以简单地通过投票法来完成，将在选择节点处获得多数投票的类作为该节点的预测值。在这种情况下，只含两种选择项的选择节点（如图7-10）没有多大意义，因为只有在两个分支都一致时才会产生一个多数投票类。另一种可能方法是平均从不同路径上所获得的概率估计，可以使用不加权的平均或是更为复杂的贝叶斯方法。

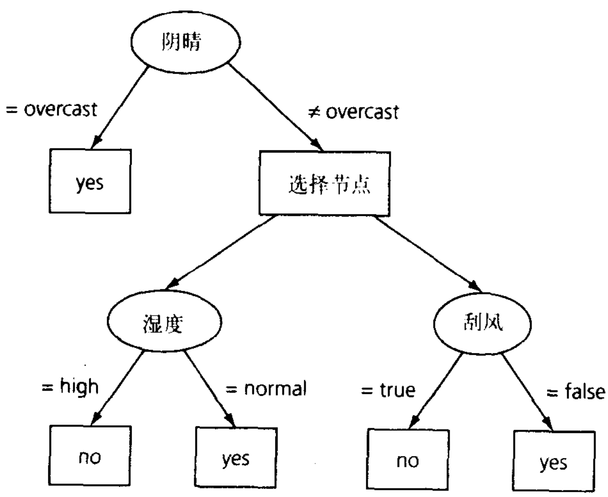


图7-10 天气数据的简单选择树

如果根据信息增益选择树中存在几处效果相似的分裂，可以通过修改现存的决策树学习

器来创建一个选择节点，生成一棵选择树。所有在某个用户指定的最好选项容许范围内的选项均可放入选择项中。在修剪时，选择节点的误差率是它的选项的平均误差率。

另一个可能方法是通过递增增加节点来扩展一个选择树。通常是采用提升算法，结果树通常称为交替式决策树（alternating decision tree）而不是选择树。这时决策节点称为分裂节点（splitter nodes），选择节点称为预测节点（prediction nodes）。预测节点如果没有决策节点增加进来便为叶节点。标准的交替式决策树能应用于二类问题，每个预测节点与一个正的或负的数值相关联。要获得对一个实例的预测，将其沿树向下过滤所有可适用的分支，并总和
329 沿途遇到的所有预测节点处的值；依据总和数是正数还是负数，来预测是哪个类别。

图7-11展示了一个简单的关于天气数据的样本树，图中正数对应于类别*玩* = no，负数对应于类别*玩* = yes。要对*阴晴* = sunny，*温度* = hot，*湿度* = normal，*刮风* = false 这些实例进行分类，将其沿树向下朝着相应的叶节点过滤，获得值 −0.255、0.213、−0.430以及 −0.331。这些值的总和是负的，因此预测出*玩* = yes。正如此例所示，交替式决策树总是在树根有个预测节点。

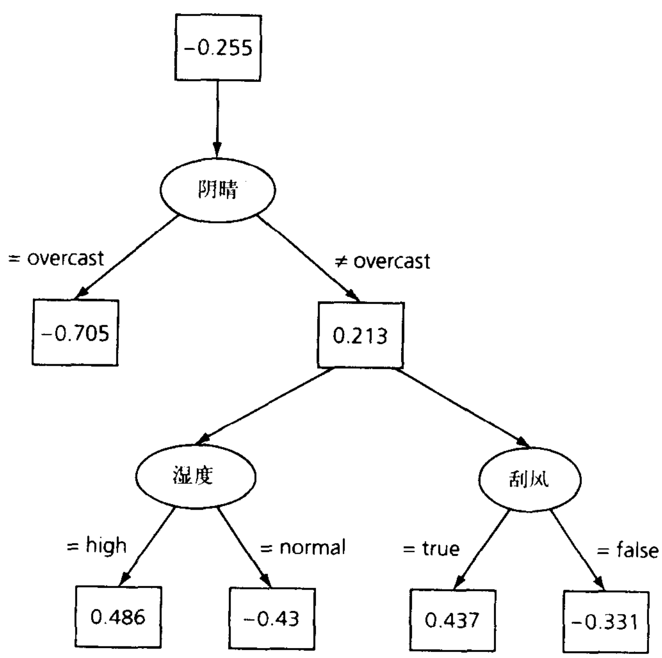


图7-11　天气数据的交替式决策树

交替式树应用提升算法来进行扩展。例如，采用基本学习器是用于数值预测的提升算法，譬如先前所述的LogitBoost方法。假设基本学习器在每轮提升循环中产生一个联合规则。那么简单地将每条规则加入到树中便能形成一个交替式决策树。与每个预测节点相关联的数值评分是从规则中获得的。但是，结果树可能很快就变得很大，因为从不同的提升循环中所得的
330 规则很可能是不同的。因此用于交替式决策树的学习算法只考虑那些通过增加一个分裂节点和两个相应的预测节点（假设是二值分裂）来扩展树中某条现存路径的规则。在标准版本的算法中，要考虑在树中每个可能位置增加节点，按照具体的提升算法所决定的性能衡量来进行节点添加。可用启发式的方法来替代穷举搜索以提高学习过程的速度。

7.5.8 Logistic 模型树

基于单个结构的选择树和交替式决策树都能获得非常好的分类性能，但是当含有许多选择节点时仍然很难解释，因为要看出一个具体的预测是如何获得的变得比较困难。然而，人们发现提升也可以用于建立一个不包含任何选择节点的十分有效的决策树。例如应用LogitBoost算法归纳出一种树，树的叶节点为线性logistic 回归模型，称为logistic模型树，并可用像6.5节所介绍的用于回归的模型树一样的方法来进行解释。

LogitBoost 执行叠加logistic回归。假设在提升算法的每轮循环中要找到一个适合的简单回归函数，它考虑所有属性，找出误差率最小的简单回归函数，并将其加入到叠加模型中。如果让LogitBoost 算法持续运行直至收敛，结果得到的是一个最大似然多重logistic 回归模型。然而，为了优化它在未来数据上的性能，通常没有必要等到收敛，这样做经常是有害无益的。可以对给定的循环次数使用交叉验证估计期望性能，当性能指标停止上升则停止程序，这样可以来决定提升的大概循环次数。

这个算法的一个简单扩展便可得到logistic模型树。当数据中不再存在任何结构可以用一个线性logistic 回归函数来建模时，终止提升程序。但是，如果将注意力限定在数据的子集上，也许仍然存在可用线性模型来匹配的某种结构，这些数据子集可以通过譬如标准决策树使用的信息增益来获得。一旦添加更多的简单线性模型也不再能获得性能提高，便分裂数据，在每个数据子集上各自重新开始提升 。这个过程将目前所生成的logistic模型分别在每个子集的数据上进行精练。在每个子集上再次使用交叉验证决定出在该子集上合适的循环次数。

递归地应用这个程序直到子集太小为止。结果树肯定会对训练数据过度拟合，可以用决策树学习的一个标准方法来修剪结果树。试验显明修剪过程非常重要。使用交叉验证法来选择合适的树大小，采用这样的策略能使这个算法生成小但非常精确的树，树的叶节点带有线性logistic回归模型。 331

7.5.9 堆栈

堆栈式泛化（stacked generalization）简称堆栈是组合多种模型的另一种不同的方法。虽然是在好些年以前就开发了，却不如装袋和提升应用广泛，一部分原因是难以进行理论分析，另一部分原因是没有一致公认的最好的实现方法，基本观点存在许多不同的应用方式。

不像装袋和提升，堆栈法通常不是用于组合同种类型的模型，譬如，一系列的决策树。相反它是应用于由不同学习算法所建的模型。假设你有一个决策树归纳器、一个朴素贝叶斯学习器和一个基于实例的学习方法，要用某个给定的数据集形成一个分类器。通常的程序是使用交叉验证法估计每个算法的期望误差率，然后从中选择一个最好的模型用于在未来的数据上做预测。难道没有更好的方法了吗？现有三个学习算法，难道不能利用三个（模型）来预测，然后将输出结果组合起来吗？

一种组合输出的方法是使用投票，就像装袋中所使用的机构。但是，如果学习方案性能相当好，（不加权的）投票才有意义。如果三个分类器中有两个所做的预测是不正确的便有麻烦了！然而，堆栈引入了替代投票程序的元学习器（metalearner）概念。投票的问题在于不清楚要相信哪个分类器。堆栈试图去学习哪个分类器是可靠的，它使用另一种学习算法即元学习器 ，来揭示怎样才能最好地组合基本学习器的输出。

元模型的输入亦称为1层模型（level-1 model），是基本模型（或称0层模型）所做出的预

测。1层实例所含的属性数量等于0层学习器的个数，属性值是这些学习器对相应的0层实例所做出的预测。当使用堆栈学习器进行分类时，实例首先被输入0层模型中，每个模型猜测输出一个类值。这些猜测值被输入1层模型，这个模型将它们组合输出最终的预测。

还存在训练1层学习器的问题。需要找到能将0层训练数据（用于训练0层学习器）转换为1层 训练数据（用于训练1层学习器）的方法。这似乎很直截了当，让每个0层模型对训练实例进行分类，将实例的真实类别附加到它们的预测结果上得到1层的训练实例。不幸的是，效果不怎么好。它会允许学习这样的规则，譬如总是相信A分类器的输出却忽略B以及C 分类器的输出。这个规则也许对于某些特定的基本分类器A、B和C是适合的，如果真是这样，或许
332 就会被学习到。但是，只是似乎适合训练数据并不意味着在测试数据上会好，因为这不可避免地会更倾向于对训练数据过度拟合的分类器，而不是那些能够做出更与实际相符的决策的分类器。

因此，堆栈不是简单地用这种方法将0层训练数据转换为1层训练数据。回顾第5章中所述的有比使用训练集上的误差率更好的方法来估计一个分类器的性能。一种方法是旁置部分实例，并用它们进行一个独立的评估。将这个方法应用于堆栈，保存部分实例作为形成1层学习器的训练实例，用其余的实例来建立0层分类器。一旦建立了0层分类器，便用它们对旁置的实例进行分类，如先前所述那样形成1层训练数据。由于0层分类器没有用这些数据训练，对它们所做的预测是无偏的，因此，1层训练数据能精确反映0层学习算法的真实性能。一旦采用这种旁置程序生成了1层数据，可再次应用0层学习器在整个训练数据上训练生成分类器，以更好地利用数据并获得更好的预测。

旁置法不可避免地剥夺了1层模型的部分训练数据。在第5章中介绍了交叉验证法作为应付这个问题出现在误差估计上的一条妙计。这种方法也可以和堆栈一起联合使用，对每个0层学习器执行交叉验证。训练数据中的每个实例正好在某个交叉验证测试折中出现一次，并将由相应的训练折所建立的0层模型对其做出的预测用于生成1层训练实例。这为每个0层训练实例生成了一个1层训练实例。当然，由于0层分类器要在交叉验证的每个折上训练，因此速度较慢，但是它能让1层分类器使用整个训练数据。

对于某个给定的测试实例，大多数的学习方法能输出每个类别的概率来替代只做类别预测。使用概率来生成1层数据能改进堆栈的性能。这和标准程序的区别只是在于每个1层名词性属性，代表由0层学习器所做出的预测类别，被几个数值属性所代替，每个属性代表由0层学习器所得的一种类概率输出。换句话说就是，在1层数据中的属性数量是类别数目的倍数。这个程序有个益处，即1层学习器能了解每个0层学习器对它所做预测的置信度，从而加强了两个层面学习之间的交流。

还有一个重要的问题，1层学习器适合使用什么样的算法呢？原则上，任何学习方案都可以用。但是，因为大多数工作已由0层学习器完成了，1层分类器只是一个仲裁者，因此选择一种相对简单的算法来达到此目的是有意义的。堆栈的发明者 David Wolpert 说“相对全局化、
333 平滑”的1层推广效果较好是合理的。简单的线性模型或在叶节点带有线性模型的树通常表现较好。

堆栈也适用于数值预测。在此情形下，0层模型和1层模型都要预测数值。基本的机构还是一样的，差别只是在1层数据的特性。在数值情况下，每个1层属性代表某个0层模型所做出的数值预测，数值型的目标值被附加在1层训练实例上来代替原先的类值。

7.5.10 误差纠正输出编码

误差纠正输出编码是用于改进多类学习问题的分类算法性能的一种技术。回顾第6章所述，有些学习算法，例如标准的支持向量机，只能用于二类问题。为了将这类算法应用于多类数据集，要将数据集分解成几个独立的二类问题，每个运行算法，再组合结果分类器的输出。误差纠正输出编码就是做这种转换的一种方法。事实上，这个方法效果如此好，即使学习算法能够直接处理多类数据集，应用这个方法也经常能获益。

在4.6节，我们学习了怎样将一个多类数据集转换为几个二类数据集。为每个类生成一个数据集，数据集复制了原始数据中的每个实例，但对类值进行了修改。如果实例的类别与所对应的数据集相关联，则类别标记为yes ，否则标记为no 。然后在每个这样的二类数据集上建立分类器，这些分类器能输出带有置信度的预测，例如类别为yes的估计概率。在分类过程中，测试实例被输入每个二类分类器，最终的类别为预测结果yes且置信度最大的那个分类器所对应的类别。当然，这个方法对分类器所得出的置信度的正确性很敏感，如果部分分类器夸大了它们的预测，总体结果将受影响。

考虑一个含4个类*a*、*b*、*c*和*d*的多类问题。数据集转换如表7-1(a)所示，yes和no分别对应为1和0。每个原始的类值被转换成一个4位的编码，每个类别对应于1位，需要用4个分类器分别预测每个位。根据这些编码来诠释分类过程，当错误的位得到了最高的置信度便发生了错误。 334

表7-1 将多类问题转换为二类问题：(a) 标准方法 (b) 误差纠正编码

类	类别向量	类	类别向量
a	1000	a	1111111
b	0100	b	0000111
c	0010	c	0011001
d	0001	d	0101010
(a)		(b)	

但并不是一定要使用如表所示的这种编码。实际上，为何每个类必须用4位编码表示也是毫无理由的。再来看表7-1(b)所列的编码，这里用7位编码来代表每个类别。应用于数据集时，要建立7个分类器而不是4个。为了看清得到的是什么，考虑对某个具体实例的分类。假设实例是属于*a*类的，各个分类器的预测（分别）是1011111。很明显，将此编码与表7-1(b)中的编码相比较，第二个分类器出错了：预测是0而非1，即no而非yes 。然而，将这个编码的预测位与关联每个类别的编码相比较，这个实例显然更接近于类别*a*。可以通过（计算）转换预测编码成为表7-1(b)中所列编码所必须改变的位数使其量化，称为汉明距离（Hamming distance），或者说对于*a*、*b*、*c*和*d*四个类别的差异分别为1、3、3和5位。这样便可以明确地定论第二个分类器出现了错误分类，并能正确分辨出*a*是实例的真实类别。

用表7-1(a)中的编码不可能实现这样的误差纠正，因为除了这4种4位编码，对于其他任何预测出的4位编码，至少会产生2个相等距离。这种输出不能“纠错”。

是什么决定了一种编码可以或者不可以纠错呢？考虑代表不同类别的编码之间的汉明距离。能够被纠正的错误数量是由任意两个编码之间最小距离*d*所决定的。编码可以保证最多纠正$(d-1)/2$个错误位，因为即使这个位数的纠正编码发生了错误，它仍然是最近的，因此能被正确地分辨出来。在表7-1(a)中每对编码之间的汉明距离是2，因此最小距离*d*也等于2，所

以能纠正的错误位数是0！然而，在表7-1(b)中，编码间的最小距离是4（实际每对编码间的距
335 离都是4）。这便意味着它能确保纠正一个错误位。

我们已可以确定好的误差纠正的一个特性：编码必须按照它们的汉明距离适当地分隔开来。由于它们构成了编码表的行，这个特性称为行分隔（row separation）。一个好的误差纠正编码所应满足的第二个要求：列分隔（column separation）。每对列之间的汉明距离必须要大，每个列和其他列的互补列之间也必须要有这么大的距离。在表7-1(b)中，7个列与其余每列（以及其余每列的互补列）分隔至少有1位。

列分隔是必要的，如果两个列是相同的（或者一个是另一个的互补列），那么对应的分类器会产生同样的误差。如果误差是关联的，换句话说，如果许多位置同时出现错误，那么误差纠正能力将被削弱。列之间的距离越大，便有可能纠正越多的错误。

对于类别数量少于4类的情形，不可能组建出一个有效的误差纠正编码，因为不可能同时获得好的行分隔和好的列分隔。例如，对于一个三类问题，只有8个可能的列（2^3），其中4个与另外4个互补。而且全为0的列和全为1的列没有任何分辨能力。只剩下3个可能的列，造成结果编码不具备误差纠正能力。（实际上，这是标准的“一个类对应一个编码”的编码方法。）

如果类别数很少，可以建立如表7-1(b)所示的穷举误差纠正编码。在k个类别的穷举编码中，列由所有可能的k位字符串组成，除掉互补列以及全0或全1的字符串。每个编码包含$2^{k-1}-1$位。编码结构如下：对应第一个类别的编码全部由1组成；对应第二个类别的编码由2^{k-2}个0紧跟着$2^{k-2}-1$个1组成；对应第三个类别的编码由2^{k-3}个0紧跟着2^{k-3}个1紧跟着2^{k-3}个0再紧跟着$2^{k-3}-1$个1组成；依此类推。第i个编码交替轮换2^{k-i}个0和1，最后一轮要少一位数。

类别数多，由于列数呈指数增长，要建的分类器太多了，因此穷举编码不可行。这时，可以应用更为复杂的方法，利用较少的列数生成一种具有好的误差纠正特性的编码。

误差纠正输出编码不能用于局部学习算法，诸如通过观察附近训练实例来预测一个实例类别的基于实例的学习。在使用最近邻分类器情形中，所有的输出位都是利用相同的训练实例所做出的预测，这个问题可以巧妙地通过使用不同的属性子集来预测每个输出位，从而解
336 除预测关联。

7.6 使用没有类标的数据

在第2章介绍机器学习过程时，将有指导的学习和无指导的学习之间加以了严格的区别，即分类和聚类。最近，研究者们开始探索位于两者之间的领域，有时称为半指导学习，目的是要进行分类，但是输入是由有类标和无类标的数据共同组成的。当然，没有带类标签的数据是无法进行分类的，因为只有类标签可以告知类别。但是，有时用一大堆无类标数据来扩展一个少量的有类标数据是很具有吸引力的。结论是无类标数据有助于类别学习。这怎么可能呢？

首先，为什么需要这样做？许多情况都呈现大量的原始数据，但是要赋予它们类别却代价很高，因为要靠人类的洞察力。文本挖掘便是经典的例子。假设要按照预先定好的类别来进行网页分类。在学院设置中，你可能对教职员工、研究生、课程信息、研究组、系科等等这些网页有兴趣。可以从大学网站上很容易地下载数千个、或者数百万个相关的网页。但是给训练数据加上类标签却是个费力的人工过程。或者，假设你的工作是要利用机器学习来识别文本中的名称，区分出人名、公司名和地名。下载几兆、或者几千兆的文本是很容易的，但是要从中挑出名称并进行分类，将其转换为训练数据，这个任务只能靠人工完成。新闻稿

件分类、电子邮件分类、学习用户阅读兴趣，应用是非常广泛的。先将文本撇在一边，假设要学习从电视新闻广播中识别某个著名人物。可以很容易录下成千上百个小时的新闻节目，但要赋予标签还是要用人工方式。在上述这些情形中，如能利用一大堆没有标签的数据，从仅有的一些有标签的实例中获得优异的性能，这将是件令人感兴趣的事，尤其是假设你是个研究生却要做赋予标签的工作！

7.6.1 用于分类的聚类

怎样利用没有类标的数据来改善分类？这里有个简单的想法。应用朴素贝叶斯法从一个小型的有类标的数据集中学习类别，然后使用6.6节中所述的EM（期望最大化）迭代聚类算法扩展应用到大型的没有类标的数据集上。程序如下：首先，利用有标签数据训练出一个分类器。第二步，将分类器应用到没有标签的数据上，根据所得类概率赋予它们类标（“期望”步骤）。第三步，使用所有数据的类标训练出一个新的分类器（“最大化”步骤）。第四步，进行迭代直至收敛。可以将此想象为迭代聚类过程，起始点和聚类标签是从有标签的数据上获 337
得的。EM程序确保在每轮迭代中找到大于或等于似然的模型参数。只能从试验中得到答案的一个关键性问题是，这些更高的似然参数估计器是否能改善分类正确率。

从直觉来看，这个效果应该不错。考虑文件分类问题。某些词组预示着类别。有些出现在有标签的文件中，而另外有一些只出现在没有标签的文件中。然而也有可能在某些文件中两者都包含了，EM 程序正是使用了这类文件，推广所学的模型从而利用了那些不在有标签的数据集中出现的词组。例如，导师（supervisor）和博士生课题 (PhD topic) 这两个词组可能都是预示了研究生主页。假设有标签的文件中只出现前面那个词组，EM迭代对模型进行推广，能对只包含后面那个词组的文件进行正确分类。

这适用于任何分类器与任何迭代聚类算法。然而，它从本质上来说是一个自引导程序，要小心确保反馈循环是正的。使用概率而非强硬的决策似乎更有益，因为它使程序慢慢地收敛而不是立即做出可能是错误的结论。在6.6节中所述的朴素贝叶斯和概率EM程序是特别适合的选择，因为它们共有相同的基本假设：属性间的独立性，或者更精确地说，对于给定类别属性之间条件独立。

当然，独立假设被普遍性地违反。甚至在前面所述的例子中，用了由两个单字所构成的词组PhD topic而在真实程序实现时，希望属性都使用单个的单词，假如我们单用PhD或topic中的任何一词，上述例子限制性就差多了。词组博士生（PhD students）可能较之研究生网页更预示着是教职员工网页；而词组研究课题（research topic）却不太有分辨力。事实是对于例子所给定的类来说，PhD和topic不是条件独立的：是它们的组合表现出了研究生网页的特性。

然而，如此结合朴素贝叶斯和EM 在文件分类领域效果不错。在某个具体的分类任务中，仅使用了少于三分之一的带有类标的训练实例以及（训练实例）五倍之多的没有类标的实例，却得到了传统学习器所能达到的性能。当有标签的实例代价昂贵而没有标签的实例可免费获得时，这是一个很好的交易。当有标签文件数量很少时，结合大量无标签的实例能大大提高分类正确率。

两种程序改进被显示能改善性能。第一种是根据试验证据而来，当存在较多有标签文件时，结合无标签实例可能会降低正确率却不能提高正确率。经手工标签的数据自然要比自动 338

标签的数据存在更少的干扰。解决方法是引入一个能减少无标签数据贡献的加权参数。这可以结合在EM的最大化步骤中，使有标签和无标签实例的加权似然最大化。当参数接近于0时，无标签文件在EM的上升面上几乎没有影响；当参数接近于1时，算法回复到原始版本，两种文件对其影响是相同的。

第二种改进是允许每个类别有几个聚类。正如6.6节中所述，EM 聚类算法假设数据是从一个含有不同概率分布的混合体中随机生成的，每个聚类对应一种分布。到目前为止，都是假设混合成员和类别之间是一一对应的。在许多情形中，这是不现实的，包括文件分类，因为大多数文件涉及多种主题。当采用每个类别含有多个聚类时，每个有标签的文件最初是以一种概率的形式随机赋予到它的每个成员中的。EM 算法的最大化步骤和以前一样，但是期望步骤被修改成不只是赋予每个实例类别概率标签，而是按照概率将其赋予每个类所含的成员。每个类别的聚类数量是一个由领域所决定的参数，可以用交叉验证来设定。

7.6.2 联合训练

使用无标签数据改善分类性能的另一种情形是在分类任务中存在两种不同且独立的观察点之时。这里再次引用经典的文件例子，这回来看网页文件，它的两个观察点是网页的内容和从其他网页到达这个网页的链接（link）。这两点都很有用但却不同：成功的网页搜索引擎使用某种秘方来利用这两个观察点。能到达另一个网页的某个链接所标注的文字为揭示另一个网页所含内容提供了线索，也许比揭示这个网页本身的内容更多，特别是当这个链接为独立链接之时。从直觉上看，被标为my adviser的链接强烈预示了目标网页是一个教职员工主页。

称为联合训练（co-training）的想法正是如此。给定少量的有标签实例，首先，为每个观察点学习一个不同的模型。在这个例子中，是一个基于内容的模型和一个基于超链接的模型。然后分别利用每个模型来赋予无标签实例类别。对每个模型，选择一个标为肯定实例且置信度最大的实例，和一个标为否定实例且置信度最大的实例，将它们加入到有标签实例队伍中。为做得更好，可通过选择一种类别实例的数量多于另一种类别，以维护有标签实例队伍中肯定和否定实例的比例。无论哪种情形，重复整个过程，在经扩展的有标签实例集上训练两个
339 模型，直到无标签实例都用完为止。

有试验证明，使用朴素贝叶斯作为学习器，这个自引导过程优于应用两个观察点的所有属性、从有标签数据上来学习单个模型。它依据从两个不同角度来看一个实例，它们是冗余的但又不完全相关联。它被提议用在不同的领域中，从分别使用视频和音频在电视新闻广播中识别名人，到带有图像、声波定位仪和距离传感器的移动式遥控装置。观察角度的独立性降低了两个臆测一致同意某个错误分类标签的似然。

7.6.3 EM和联合训练

对于含有两个真正独立的属性子集的数据集，试验显示，如前所述，联合训练的效果比使用EM策略更好。而将二者组合起来形成改进的联合训练法称为co-EM，能获得更好的性能表现。联合训练法训练出两个代表不同观察点的分类器A和B，利用它们在无标签的实例中挑选被预测为最肯定的和最否定的实例，并往训练队伍中添加这些新实例，新实例的数量极少而且类别是确定的。另一方面，Co-EM在有标签的实例上训练A，然后用它根据类概率赋予所有无标签实例以类标签；接着在有标签实例以及由A 暂定标签的无标签实例上训练分类器

B；然后（再用B）根据概率重新赋予所有实例以类标签，并用于训练分类器A；重复迭代过程直至分类器收敛。这个程序效果始终比联合训练法更好，是因为它没有被分类器A 和B所生成的类标签所束缚，而是在每轮迭代中重新估计它们的概率。

co-EM的适用范围如联合训练一样，仍然受限于多种独立观察点的要求。但是有试验证明即使不能自然地将属性分裂成独立的观察点，通过制造这样的分裂并使用联合训练，或者更好使用co-EM分裂数据，也能从中获益。甚至采用随机分裂时，似乎也能奏效；策划分裂使属性集达到最大独立，性能肯定能获得提高。这样做为什么会有效呢？研究者们猜测，这些算法能获成功，部分原因是分裂更加强了它们对潜在分类器所做的假设。

没有任何特别的理由要限定基本分类器为朴素贝叶斯法。支持向量机可能代表了当今用于文本分类最为成功的技术。但是，对于EM迭代法，分类器必须用概率来赋予数据类标；还必须能使用经概率加权的实例来做训练。支持向量机很容易被调整以适应这两点。在6.5节的局部加权线性回归（6.5节）中介绍了怎样调整学习算法来处理加权实例。从支持向量机中获 340
取概率估计的一种方法是用一维logistic模型来匹配输出，正如4.6节中所述的在输出上有效使用logistic 回归。在文本分类中使用co-EM和支持向量机（SVM）分类器获得了极好的效果。它的性能优于其他版本的SVM，而且对于各种比例的有标签和无标签实例都能运用自如。

联合训练和EM的想法，特别是co-EM将它们组合起来的观点，是令人感兴趣的、启发式的思考，并有惊人的潜力。但到底是何原因使它们如此有效仍存争议、还不能被理解。这些技术是当前研究的课题：还未纳入机器学习的主流而是作为数据挖掘的实践工具。

7.7 补充读物

属性选择已经在模式识别领域中被探索了几十年了。例如反向消除法是在20世纪60年代初引入的（Marill 和 Green，1963年）。Kittler（1978年）对用于模式识别的属性选择算法进行了考察。最佳优先搜索和遗传算法是标准的人工智能技术（Winston，1992年，Goldberg，1989年）。

添加新属性会使决策树学习器性能变差的试验结果是由John在1997年报告的，他对属性选择进行了适当的解释。找到能独特地划分实例的最小属性集的思想方法是由Almuallin 和Dietterich（1991年、1992年）提出的，并在1996年由Liu和Setiono进行了深入开发。Kibler、Aha（1987年）以及Cardie（1993年）都研究了用决策树算法来为最近邻学习确定属性；Holmes 和Nevill-Manning（1995年）使用1R 将属性排序用于选择。Kira 和Rendell（1992年）使用基于实例的方法来选择属性，产生一种方法称为递归消除属性法（RELIEF）。Gilad-Bachrach 等人在2004年展示了如何修改此方法来更好处理重复属性。基于关联的属性选择方法是由Hall 在2000年开发的。

使用包装的方法进行属性选择要归功于John 等（1994年）以及Kohavi和John（1997年）。Vafaie和DeJong（1992年）以及Cherkauer和Shavlik（1996年）将遗传算法应用于包装结构中。选择性朴素贝叶斯学习方法应归于Langley和 Sage（1994年）。Guyon等（2002年）展现并评估了递归属性消除方法结合支持向量机的方案。竞赛搜索法是由Moore 和 Lee（1994年）开发的。 341

Dougherty 等（1995年）简单地叙述了有指导和无指导的离散，并将试验结果与基于熵的等宽装箱和单规则（1R）方法进行了比较。Frank 和 Witten（1999年）描述了使用经离散的

排序信息的效果。用于朴素贝叶斯的k区间离散是由Yang 和Webb （2001年）提议的。基于熵的离散方法包括MDL的停止条件是由Fayyad 和 Irani （1993年）开发的。使用χ^2测试的由下往上的统计方法应归功于Kerber （1992年），Liu 和 Setiono （1997年） 介绍了由此法延伸而获的一种自动判定重要性水准的方法。Fulton 等（1995年）探索了将动态规划应用于离散，并得到一个不纯函数的二次时间边界 （例如熵） 和一个基于误差离散的线性边界。用来说明基于误差离散的弱点的例子是根据Kohavi 和 Sahami （1996年） 改写的，是他们首先清楚地发现这个现象的。

主分量分析法是一种标准技术，大多数统计教科书中都有介绍。Fradkin 和 Madigan（2003年）分析了随机投影的性能。Witten 等（1999年）中介绍了TF × IDF 衡量尺度。

使用C4.5过滤训练数据的试验是由John（1995年）展示的。Brodley 和 Friedl （1996年）研究了一个更为保守的方法，使用几个学习算法生成一个共识过滤器。Rousseeuw 和 Leroy（1987年）介绍了统计回归中的孤立点侦察，包括最小二乘方法；他们还展示了图7-6的电话数据。Quinlan（1986年）发现，消除训练实例属性上的干扰会由于相似的干扰测试实例而降低分类器的性能，特别是干扰度较大时。

组合多个模型在机器学习研究领域是一个热门研究课题，已有很多相关的出版文献。术语装袋 （即自引导整合）是由Breiman（1996年）提出的，他从理论角度、试验角度对用于分类和数值预测的装袋特性都做了研究。Domingos（1999年）介绍了MetaCost算法。Dietterich（2000年）对随机化进行了评估，并将其与装袋技术和提升技术做了比较。Bay（1999年）提议使用随机化方法对最近邻分类器进行合成学习。Breiman（2001年） 介绍了随机森林方法。

Freund 和 Schapire （1996年）开发了AdaBoost.M1的提升算法，并获得了这个算法性能表现的理论边界。以后，他们又应用边差的概念改进了这些边界 （Freund和Schapire，1999年）。Drucker （1997年） 修改了AdaBoost.M1使其能用于数值预测。LogitBoost算法是由Friedman 等（2000年）开发的。Friedman （2001年） 介绍了如何使提升在有干扰数据时更
342 具恢复力。

Domingos（1997年） 介绍了怎样利用人工训练实例从一个合成分类器中获得易于说明的单个模型。贝叶斯选择树是由Buntine （1992年）介绍的，Kohavi 和 Kunz （1997年） 将多数投票法和选择树结合应用。Freund 和 Mason （1999年）引入了交替式决策树；Holmes等（2002年）对多类交替式决策树进行了试验。Landwehr 等（2003年）用LogitBoost算法开发了logistic 模型树。

堆栈式泛化是由Wolpert（1992年）初创的，他在神经网络文献中提出这个想法，Breiman（1996年）将其应用于数值预测。Ting和Witten（1997年）通过试验比较了不同1层模型，结果发现使用简单的线性模型效果最好；他们还证明了利用概率作为1层数据的益处。他们还对堆栈和装袋的组合进行了探索。

使用误差纠正输出编码进行分类的思想在Dietterich和 Bakiri（1995年） 发表后得到了广泛的认可；Ricci和 Aha（1998年）展示了怎样将此编码运用于最近邻分类器。

Blum和Mitchell（1998年）是使用联合训练的倡导者，并为从不同的独立观察点使用有标签和无标签数据开发了一个理论模型。Nigam 和 Ghani （2000年） 分析了联合训练的有效性和可用性，并将其与传统的使用标准EM 方法来填补残缺值联系起来。他们还介绍了co-EM算

法。Nigam等（2000年）在用于分类的聚类这节中，展示了EM聚类算法是怎样利用无标签数据来改善由朴素贝叶斯法建立的原始分类器的。当时，联合训练和co-EM主要是应用于一些小型的二类问题；Ghani（2002年）使用误差纠正输出编码来解决带有多个类别的多类问题。Brefeld和 Scheffer（2004年）扩展了co-EM法，使用支持向量机而非朴素贝叶斯法。Seeger（2001年）怀疑这些新算法与传统算法相比，是否真的能带来一些优势。 343

第8章　继续：扩展和应用

机器学习是从数据中挖掘知识的一项新兴技术，许多人开始认真地审视这项技术。我们不想过度吹嘘，我们所了解的机器学习并非是用来干大事的，比如未来的自动机器仆佣、关于认知的哲学迷题、有关超自然的自由意志的话题、关于智慧从何而来的进化论或者是神学的问题、语言学习方面的辩论、关于儿童成长方面的心理学理论、或者关于智能为何物、又是如何工作的认知解释等。机器学习在我们心目中要平凡得多，机器学习是能从数据中推断出结构的那些算法、以及验证这些结构的方法。这些算法既不深奥也不复杂，但它们也不是显而易见的，或是不重要的。

用发展的眼光来看，最大的挑战在于应用。机会俯拾皆是。哪里有数据，哪里就有可以学习的东西。当数据过多、人们自身的脑力无法承受时，学习的手段就必须是自动的。但是灵感
345 是绝不可能“自动”产生的！应用方法不可能来自计算机程序，也不可能来自机器学习专家，或是数据本身，只能来自和数据以及问题起源打交道的人。这正是我们编写本书，以及第二部分叙述的Weka系统的目的所在，让那些非机器学习专家的人们有能力将这些技术运用到日常工作中去解决问题。思想方法是简单的，算法就在这里，其余的就要看读者自己了！

当然，这项技术的发展远没有完成。机器学习是一项热门研究课题，新的思想和技术还在不断涌现。为了提供读者一些有关研究前沿的范围和研究种类，我们来关注一下当今数据挖掘领域的一些热门课题，以此来结束本书的第一部分。

8.1　从大型的数据集里学习

当今工商企业和科研机构中盛行使用大型数据库，使得机器学习算法必须能在大型数据集里工作。任何算法要应用到非常大的数据集，存在两个关键问题：空间和时间。

假设数据过大，无法存储在主存储器中。这时如果学习方案采用递增模式，在产生一个模型时，每次只处理一个实例，就不会造成困难。可以从输入文件中读取一个实例，更新模型，然后继续读下一个实例，以此类推，在主存储器中永远无须储存一个以上的实例。一般而言，得到的模型和数据集相比比较小，可供存储空间量不对所形成的模型有严格的限制。朴素贝叶斯方法是这类算法的出色的例子；决策树归纳以及规则学习方案也有使用递增模式的。不过，本书所探讨的部分学习方法的递增算法尚未开发。其他方法，例如基本的基于实例的方法和局部加权回归法在预测时需要使用所有的训练实例，如此一来，则要利用精细的储存和索引机制，只将数据集中使用最频繁的部分保存在存储器中，并且能对文件中的相关实例进行快速访问。

在大型数据集里应用学习算法的另一个关键的方面是时间。如果学习时间不是与训练实例的数量成线性（或者几乎成线性）度量关系，那么处理非常大的数据集最终将是不可能的。在某些应用中，属性数量是一个关键因素，只有那些与属性数量成线性度量关系的方法才能被接纳。或者，预测时间也可能成为关键问题。幸运的是，许多学习算法在训练和测试中度
346 量都极其出色。例如，朴素贝叶斯法的训练时间与实例数量和属性数量都成线性关系。从上

而下的决策树归纳法，如6.1节中所见，训练时间和属性的数量成线性关系，如果树呈现枝叶繁盛，则与实例数量成对数线性关系（假如没有使用子树提升；如果用了，再要添加一个对数因素）。

如果数据集过大，以致某一种具体算法无法应用，有三种办法可以使学习变得可能。第一种不太重要，即不把学习方案运用到整个数据集中，而是仅在一个较小的子集中训练。当然，这样采用二次取样（subsampling）会造成信息遗失。不过，由于在覆盖到所有的训练数据之前，所学到模型的预测能力往往早已达到最高点，此种损失或许可以忽略不计。如果是这种情形，很容易得到证实，可通过在一个旁置测试集上观察测试由不同大小的训练集上所获模型的性能表现。

这种行为模式称为收益递减法则（law of diminishing returns），可能会发生是由于学习问题简单，因此少量的训练数据足以学到一个精确的模型。另一种可能是学习算法也许无法抓住潜在领域的详细结构。在一个复杂领域中运用朴素贝叶斯法时，往往可以观察到这样的情形，额外的训练数据或许不能改善模型的性能表现，却能使决策树模型的正确率继续上升。在这种情况下，如果主要目标是预测的性能表现，当然应该换用更加复杂的学习算法。但要注意过度拟和！不要在训练数据上进行性能的评估。

并行化（parallelization）是另一种减少学习时间复杂度的方法。想法是将问题分裂成几个小部分，每个部分用一个单独的处理器来解决，然后将结果合并。为达此目的，需要建立一个并行化的学习算法。有些算法天然地适用于并行化方法。例如最近邻方法，可以通过将数据分裂成几个部分，让每个处理器在它那部分的训练集中找到最近的邻居，从而轻易地在几个处理器间进行分配。决策树学习器可以通过让每个处理器建立完整树的一个子树而进行并行化。装袋和堆栈（虽然提升法不是）是天然的并行算法。然而，平行化只能做部分补救，因为处理器的数量是固定的、无法改善算法的渐近时间复杂度。

在大型数据集上应用任何算法的一种简单方法是将数据分裂成有限大小的数据块（chunk），在每个数据块上学习模型，再利用投票或取平均值的方法将结果组合起来。无论是并行的类似装袋的方法或是有序的类似提升的方法都可以用于此目的。提升法具有的优势可以根据先前数据块上学习所获的分类器来对新的数据块进行加权，从而在各个数据块之间传递知识。在两种情况下，存储耗费按照数据集的大小成线性增长；因此对于非常大的数据集来说进行某种形式的修剪是必要的。这可以通过将一些验证数据放置一边，只往分类器委员 347
会中添加能使委员会在验证集上的性能获得提高的新模型。验证集也可用于确定合适的数据块尺寸，这可采用几种不同大小的数据块，平行地运行算法，并且关注它们在验证集上的性能表现。

要使学习方案能够对付非常大的数据集，最好的但同时也最具有挑战性的方法是建立计算复杂度较低的新算法。在有些情况下，要获得低复杂度的算法是不可能的。处理数值属性的决策树学习器就属于这类情况。它们的渐近时间复杂度受控于数值属性值的排序过程，对于任何给定的数据集，至少要进行一次这样的数据处理过程。然而，有时可以利用随机算法以获得接近正确的解决方法，而所需的时间耗费却少得多。

背景知识可以大大减少学习算法必需处理的数据量。将背景知识纳入考虑范畴之后，大型数据集的绝大多数属性可能就显得不相关了，这取决于哪个属性是类属性。通常，将要交给学习方案的数据预先进行一番仔细处理是值得的，并且将手头已有的任何关于学习问题的

信息加以最大的利用。如果背景知识不充足，利用在7.1节中讨论的属性过滤算法常常可以大大减少数据量，也许是以预测性能表现的稍许损失为代价。其中有些方法，例如，用决策树或1R学习方案进行属性选择与属性的数量成线性关系。

想给读者感觉一下在普通的微机上直接应用机器学习算法所能处理的数据总量，将决策树学习器J4.8运行在一个拥有600 000个实例的数据集上，它有54个属性（10个数值型，44个二值属性）以及一个含7个值的类。使用一台带有奔腾4中央处理器、2.8GHz时钟、带“即时编译器”的Java虚拟机。载入数据文件、用减少误差修剪法（reduced-error pruning）建立决策树并对所有训练实例进行分类，用了40分钟时间。这棵决策树有20 000个节点。这个算法实现程序是用Java编写的，而执行一个Java程序常常要比执行一个相应的用C语言编写的程序慢几倍，这是由于Java字节码必须先翻译成机器编码方可实施（根据我们的经验，如果虚拟机采用即时编译器，这个差别常常是三到五倍）。

现今有的数据集绝对称的上大型（massive）这个形容词。比如，天体物理、核物理、土
348 壤学以及微生物学方面的科学数据集往往以成百个十亿字节计，甚至是万亿字节。包容金融交易记录的数据集同样如此。将机器学习的标准程序完整地应用到这类数据集中是一个非常具有挑战性的课题。

8.2 融合领域知识

本书自始至终强调在进行实际数据挖掘工作时，十分重要的是了解数据。专业领域的知识对于（挖掘）成功是绝对必要的。数据的数据常称为元数据（metadata），机器学习的一个前沿就是改进学习方案，使学习方案能将元数据以有用的方式纳入考虑范围。

无须寻觅如何运用元数据的实例。在第2章中，我们将属性分为名词性属性和数值属性两大类。我们也注意到或许有更好的分类定义。假如属性为数值的，意味着存在着一种序列，但零点的存在时有时无（对于时间区间存在，对于日期则不存在）。序列还可能是非标准的，角度的序列不同于通常的整数序列，因为360°等同于0° ，180°和 - 180°乃至等同900°。离散方案假设正常的线性排序，如同用于数值属性的学习方案，但将其扩展到循环排序也属常规问题。分类数据也可能被排列。想象一下如果没有传统的字母顺序，日常生活将会是多么困难（看一下香港的电话号码本，就会知道这是一个多么有趣而非同小可的问题了！）。日常生活的节奏反映在循环排序中：星期中的天数，年份中的月份数。更复杂的事务中有许多其他类型的排序，譬如在子集上的局部排序：子集A可能包括子集B，或者子集B包括子集A，或者谁也不包括谁。扩展普通的学习方案，能将此类信息以令人满意的常规方式来考虑，还有待进一步的研究。

元数据经常包含了属性之间的关系。明显地有三种关系：语义关系，因果关系及函数关系。两个属性间的语义关系意味着如果第一个属性被包括在某一条规则中，那么第二个属性也应该被包括在其中。如此说来，将其作为一个前提，就是这两个属性只有在一起才有意义。例如，在我们分析过的农业数据中，称为产奶量的属性衡量一头奶牛产多少牛奶，调查的目的意味着这个属性与其他三个属性存在语义关系：奶牛标识（cow-identifier），牛群标识（herd-identifier）和牧场主标识（farmer-identifier）。换句话说，产奶量的具体值只有在综合
349 考虑这些情况的前提下才能被理解，这包括产奶的那头奶牛，这头奶牛进一步联系到某个已知牧场主所拥有的特定牛群。语义关系当然是取决于具体问题的，它们不仅仅取决于数据集，

也取决于你要用数据集去干什么。

当一个属性引发另一个属性时，因果关系就产生了。在一个试图预测某个属性的系统中，而这个属性是由另一个属性引起的，我们知道这另一个属性必须被包括进来预测才有意义。例如，上面所述的农业数据中存在着一条从牧场主标识、牛群标识到奶牛标识的链条，这根链条继续延伸到那些经过度量的属性，如产奶量，一直到记录了牧场主是否拥有或出售了那头奶牛的那个属性。学到的规则应该能识别这条链上的依存关系。

函数依存关系存在于许多数据库中，为了对数据库中的关系进行正常化，数据库的建立者试图识别出它们。当从数据中学习时，某个属性对另一个属性的函数依存关系的重要性在于，假如后者在规则中已被使用了，就没有必要再考虑前者了。学习方案经常重新发现这个已被知晓的函数依存关系。这不仅是在产生无意义的，或是同义反复的规则，而且一些更令人感兴趣的模式也许会被函数关系所模糊。然而，人们在自动数据库设计上已经做了许多工作来解决从样本查询导出功能依存的问题，所开发的方法应该对于清除学习算法所产生的同义反复的规则是有用的。

当使用我们已经遇到过的任何学习算法来做归纳时，将这些元数据或者先验领域知识纳入考虑范围，看上去并不构成任何大的技术上的挑战。唯一真正的问题而且是个大问题，是如何将元数据以一种概要的、容易理解的方法来表达，使得人们能生成这些元数据，并为算法所使用。

使用如同机器学习方法所生成的表示法来表达元数据知识是有吸引力的。我们将重点放在作为这项工作标准的规则上。这些指明元数据的规则与该领域的先验知识相关。给出训练样本，可以用以前介绍过的某个规则归纳方案来获得其他的规则。通过这种方法，这个系统也许可以将“经验”（来自样本）和“理论”（来自领域知识）结合起来。它还可以确认和修正这些建立在实践证据上并已被结合进来的知识。简单地说，使用者告诉系统他或她所知道的，给它一些样本，系统就能自己找出其余的!

为了有效灵活地利用以规则表达的先验知识，系统必须能执行逻辑推导。不然，知识必须精确地以适当的方式表达以使学习算法能利用它，这在实际运用中可能过于强求了。考虑有因果关系的元数据：如果属性A引发B而属性B引发C，那么我们希望系统能演绎出A引发C，而不是明确地陈述事实。虽说在这个简单的例子中明确阐述这个新的（A引发C——译者注） 350
事实不会有什么问题，但在实际中，由于元数据的广泛，希望系统使用者能表达出他们先验知识的所有逻辑后果是不现实的。

组合从事先确定的领域知识中演绎出的结果以及从训练样本上所获的归纳结果，看上去是元数据的一个灵活的容纳方式。在一种极端情况下，当样本缺乏（或不存在）时，演绎是一种主要（或唯一）的产生新的规则的方式。在另一种极端情况下，当样本丰富而元数据缺乏（或不存在）时，本书叙述的标准机器学习技术就起作用了。实际情况介于这两种情况之间。

这是一个引人注目的现象，在3.6节中提到的归纳逻辑编程法提供了一种使用正式的逻辑语言的陈述来清楚地阐明领域知识的一般方法。但是，目前的逻辑编程方法在实践环境里存在严重的缺陷。它们较为脆弱缺乏活力，而且计算量过大，几乎完全不可能运用于任何实际大小的数据集中。也许这源自它们使用一阶逻辑（first-order logic）的事实，就是说，它们允许将变量引入规则中。我们所见过的机器学习方法的输入和输出以属性和常量来表示，使用命题逻辑而没有变量，极大地减少了搜索空间并避免了循环工作和终止的困难。有人热望采

用简化的推理系统来避免完整逻辑编程中的脆弱和计算无法实现这两个问题。另一些人坚信6.7节中介绍过的贝叶斯网络的一般机构，在这网络中因果限制可以在网络的最初结构中表达出来，并自动假设和评估隐藏的变量。观察一下允许表示不同种类的领域知识的系统是否会得到广泛应用是非常有意思的。

8.3 文本和网络挖掘

数据挖掘是在数据中寻找模式。类似地，文本挖掘就是在文本中寻找模式，它是一个分析文本、并从中提炼出有助于某个特定用途的信息的过程。与本书里一直讨论的数据相比，文本是无结构的、没有一定形态的、并且是难以处理的。毋庸置言，在当代西方文化里，文本是交换信息的最有效的工具。从其中提炼有用信息的动机是极有吸引力的，即便只是部分成功。

文本挖掘和数据挖掘表面的相似掩藏了它们实际的不同。在第1章中将数据挖掘定义为从
351 数据中将暗藏的、以前所不知道的、但却是潜在有用的信息提炼出来。在文本挖掘中，所要提取的信息却是清楚明了地陈述在文本中的。一点都没有隐藏，大多数作者都是经过苦苦挣扎而要确保清楚、不含糊地表达他们自己的思想。从人的思维角度来看，"以前所不知道的"的含义只能是由于时间限制使人无法靠他们自己来阅读文本。问题在于信息传达的方式无法适用于自动处理。文本挖掘试图将信息以一种适合计算机，或是没有时间阅读整个文本的人群来消化理解的形式呈现出来。

对数据挖掘和文本挖掘的一个共同要求是所提炼出的信息必须是潜在有用的。一方面，这意味着可以有所作为（actionable），可以为某些自动采取的行动提供一个基础。从数据挖掘的角度上看，这个注解可以用一个相对独立于领域知识的方式来表达：可以有所作为的模式就是那些能对相同来源的新数据做出并非无关紧要的预测的模式。可以用成功与失败的累计数来衡量性能、可以应用统计技术在同样的问题上比较不同的数据挖掘方法，等等。然而，在许多文本挖掘问题上，要想以独立于手头某个具体领域的方式，来为"可以有所作为"做一个定义是极其困难的。这使得要寻找一个公正、客观的衡量成功度的方法变得困难重重。

正如本书所强调的，"潜在有用"在实际数据挖掘中常常有另外一种解释：成功的关键在于所提炼出的信息必须是可以理解的，即可以帮助诠释数据的。这对于最终结果是用于被人脑理解而非（或者同时）用于自动操作都是必须的。这条标准不那么适用于文本挖掘，因为和数据挖掘不同，（文本挖掘的）输入本身是可以理解的。有着可以理解的输出的文本挖掘等同于对一个大文本的重要部分所做的总结，即文本归纳（text summarization）。

我们已经遇到过一个重要的文本挖掘问题，文件分类，即每个实例代表一个文件而实例的类是文件的主题。文件的性质是由文件中出现的词汇所决定的。每一个单词的出现或失踪都可以看作是一个布尔属性，或者文件可以当作一个词汇包而非词汇集，词汇包将单词出现的频率纳入考虑。在4.2节，学习如何扩展贝叶斯网络成词汇包代表，从而产生算法的多项式版本。

不同单词的数量当然是极其众多的，它们中的大多数对于文件分类都不是很有用的。这是一个经典的属性选择问题。有些词，例如功能词汇常常称为停止词（stopwords），可被先验消除，这些词虽说出现频繁但它们的总数并不多。其他单词出现次数过于稀少，似乎也不会
352 对分类有用。荒谬的是，单词出现几率不频繁是普遍现象，一般文件或文集中近一半的词只

出现一次。毋庸置言，将停止词都去除后仍然存在这么多数量的词汇，因此有必要使用7.1节中所述的方法来进行更深入的属性选择。另一个问题是单词包（或单词组）模型忽略了单词次序和上下文效果。有效果很好的例子通过侦察出通用词组并将其看作单独个体来处理。

文件分类是有指导的学习，类别是已知的，每个训练文件都事先赋予了类别。无指导的问题版本称为文件聚类。这时类不是事先确定的，但寻找出的是同类的文件。文件聚类可以帮助信息检索，它在相似的文件之间建立联系，一旦其中的一个文件被认定和所查询的相关时，相关的文件便依次被检索出来。

文件分类的应用是相当多的。一个相对较容易的分类任务语言识别（language identification）为国际文件库提供了一个重要的元数据。一个简单而语言识别效果不错的表达方式是，由文件中出现的n-grams、或者n个连续的字母序列所构成的文件轮廓（profiles）来为每个文件定性。出现次数最为频繁的300个字母序列或称n-grams是和这种语言密切相关的。一个更富挑战性的应用是作者归属（author ship ascription）问题，应用于当一个文件的作者不确定、必须从文本中来猜测时。这里起作用的是停止词，而非内容词汇，因为停止词的分布取决于作者而与主题无关。第三类问题是从一个受控的可能词组的词汇表里向文件赋予关键词组（assignment of key phrases），给定大量加了标签的训练文件，标签也源自词汇表。

另一类常见的文本挖掘问题是元数据提炼（metadata extraction）。在前面元数据称为数据的数据，在文本领域里这个术语一般是指作品的显著特征，例如它的作者、题目、主题分类、课题名称以及关键词。元数据是一种高度结构化的（因此也是可以有所作为的）文件归纳。元数据这个概念常常被扩展，可以包含代表世界上的事物或“实体”的单词或词组，这便产生了实体提炼（entity extraction）这个术语。普通的文件中充满了这类术语，如电话号码、传真号码、街道号码、电子邮件地址、电子邮件签名、文章概要、内容目录、索引文件清单、表格、图形、标题、会议声明、网址以及其他许许多多。此外，还有数不胜数的特定领域的 353
实体，例如国际标准书号（ISBN）、股票代码、化学结构式以及数学方程等。这些术语当作是词汇表中的单独词汇项，如果它们能被识别出来，许多文件处理工作可以大大地改进。它们对于文件搜索、内部链接、以及在文件之间的互相索引工作上都有帮助。

文本类的实体是如何被识别的呢？机械的学习方法，即查字典，固然是一个办法，特别是在手头有现成的资料，如人名或组织清单、来自地名字典的地址信息、缩简写字典。另一个办法是利用人名和简称的大写和标点符号模式；称谓（女士 Ms.）、后缀（Jr.）以及贵族的前缀（von）；罕见外国人名的语言统计等等。对于人工结构，如信息位置的表示方法（URL）可以用正规统一的表达式来满足；可以记录下清楚的语法来识别日期和金钱数量。即便是最简单的任务也是提供了学习如何应付实际生活中千变万化的文件的机会。举个例子来说，还有什么比查询表中的名字更为简单的任务？但是利比亚领导人卡扎菲的名字在国会图书馆所收集到的文件中居然有47种不同的表达形式！

许多篇幅短小的文件对某个特定的主题或者事件进行描述，将多个实体组合成更高级别的合成体来代表文件的完整内容。识别合成体结构的任务称为信息萃取（information extraction），这个合成结构经常可以表示为含有许多槽的模版，每个槽里填有单条结构信息。一旦实体被识别出来后，文本被解析以决定实体之间的关系。典型的萃取问题要求寻找预先决定的一组命题的预测结构。这些通常是简单的解析技术，譬如有限状态语法（finite-state grammar）就能捕获了，虽然由于不明确的代词、介词短语以及其他一些修饰语可能使情况变

得复杂一些。机器学习技术也用于信息萃取，寻找提取模板槽内填充内容的规则。这些规则也许是以模式-行为（pattern-action）的形式存在，模式表达了对填充物以及上下文中词汇的限制条件。这些限制条件可能关系到词汇本身、它们的词性标签以及它们的语义类别。

将信息萃取再深入一步，被提取的信息可以在接下来的步骤中用来学习规则，不是关于如何提取信息的规则，而是为文本本身的内容定性的规则。这些规则可能会从文本的其余部分中来预测某个槽中的填充内容。在那些严格受限的情况中，例如在互联网上张贴与计算机相关的职位，根据规则判定的质量，建立在少量人工组建的训练样本基础上的信息萃取能与完全由人工组建的数据库抗衡。

国际互联网是一个文本的大型储藏室。由于它包含了明确的结构，几乎全部与普通的
354 “简单”文本不同。有些结构属内部结构，标明了文件的组织结构或格式；其他则属外部结构，定义了文件之间的明确的超文本链接。这些信息资源为网络文件挖掘提供了额外的支持。网络挖掘（web mining）与文本挖掘类似，只是还拥有额外信息所带来的优势，经常可以利用标题目录和其他网络上的信息来改善结果。

包含关联数据的互联网资源，如电话号簿或产品目录使用超文本结构语言（HTML）格式命令向网络用户清楚地展示它们所包含的信息。然而，要从这种资源里自动地将信息提取出来是相当困难的。为达此目的，现有的软件系统使用称为包装器（wrappers）的简单分析模块来分析网页的结构并提取所需的信息。如果手头有编写好的包装器，这就成了普通的文本挖掘问题了，因为信息可以从固定的、预定结构的网页中，使用算法被提炼出来。但是网络很少遵守规则。它们的结构是多变的，网站也在发展。在人看来并不显著的错误会使自动提炼程序完全错误。当变化发生时，人工调整包装是一件令人痛苦的事情，需要投入思考当前程序并对其进行修补而又不能引起其他地方遭受破坏。

现在来看包装归纳（wrapper induction），自动从实例中学习包装器。输入是一个网页的训练集，带有元组、元组代表从每个网页上提取出的一系列信息。输出是一系列规则，这些规则是通过分析网页来提炼元组的。例如，它也许会寻找某些网页设计者用于分隔关键信息项目的HTML分隔符，如分段符（*<p>*）、输入序列符（*<li>*）或粗体符（*<b>*），并学习显示信息的序列。完成这项工作可通过对所有分隔符选择进行迭代循环，当遇到统一的包装即停止。然后识别只是根据最少的提示，对输入中无关文字和记号做一些防御。或者，也可以按照5.9节最后所提的伊壁鸠鲁建议来寻找一个强壮的包装器，采纳多种提示以备意外变化。自动包装归纳的一大优点是，当错误由于格式变化而引起时，只要简单地将它们添加到训练数据中，再重新归纳一个将其考虑进去的新包装。包装归纳减少了当出现小变化时识别中存在的问题，并使结构发生根本变化时，产生新的提炼规则容易多了。

一种称为语义网络（semantic Web）的新发展试图能让人们以一种清楚呈现信息结构和语
355 义的方式来公布信息，以便信息能够被再利用而不只是用于读取。这会使包装归纳变得多余。但是，如果使用语义网络所要求的人工结构不提及遗传下来的大量篇幅的页面内容，很可能会使对自动信息结构归纳的要求增加。

文本挖掘，包括网络挖掘作为一项新兴技术，由于它的新颖和内在的困难，还处在一种不稳定的状态，也许和机器学习在20世纪80年代中期的状态非常类似。它究竟涵盖哪些问题还没有达成真正共识。广而言之，所有自然语言处理都处在文本挖掘的范围之内。由于文本挖掘任务对于所考虑的具体文本高度敏感，通常很难提供一种普遍适用、有意义的评估方法。

自动文本挖掘技术在能超越人类的这项能力之前，即使是不涉及任何特殊的领域知识，只是从如山的文件堆里获取信息，也还有一条漫长的道路要走。但它们会走这条漫长的道路，因为需求是极广大的。

8.4 对抗情形

机器学习的一个重要应用是垃圾邮件的过滤。在我们写作本书时（2004年末），不想要的邮件是一个极为令人烦恼的问题，也许当你们读此书时这个恶魔已经被赶走了，或者至少被驯服了。乍看上去垃圾邮件过滤是一个标准的文件分类问题：根据它们所包含的文字内容，在大量训练数据的指导下，将文件分成“火腿”和“垃圾”两大类型。但它不是一个标准的问题，因为它有对抗性的一面。被分类的文件不是从那个无法想象的、巨大的所有可能的文件集中随机抽取的，它们包含那些经过精心包装可以逃避过滤程序、被特别设计来击败系统的电子邮件。

早期的垃圾过滤器简单地将包含诸如暗示性、钱财及庸医的典型垃圾邮件字眼的讯息清除掉。当然，许多诉讼来往（邮件）涉及到性别、金钱和药物，因此必须有所权衡。所以过滤器的设计者们运用贝叶斯文本分类方案，在训练过程中力求找到一个适当权衡。垃圾邮件的制造者们很快调整了策略，利用拼写错误将那些典型字眼隐藏起来；用合法的文本包装它们，也许是在白色的背景下用白色的字打印的，使得只有通过特别的过滤器才可以看到；或者简单地将垃圾文本放在其他地方，放在图像或URL上，绝大多数邮件阅读器会自动下载它们。

很难客观地比较垃圾邮件侦察算法，这个事实使问题变得更为复杂了，虽然训练数据很多，但隐私问题阻碍了将大量有代表性的邮件公之于众。并且还有强烈的时间效应，垃圾邮件快速地改变特性，使交叉验证之类的敏感的统计测试无效。最后，“坏蛋”也可以利用机器 356
学习。例如，如果他们能够得到过滤器所阻止的和所放行的样本，便可以用这些作为训练数据学习如何逃避过滤。

不幸的是，在今天的世界里还有许多其他对抗性的学习情形的例子。和垃圾邮件问题密切相关的是搜索引擎垃圾：网站试图误导互联网搜索引擎，将它们置于搜索结果清单中显眼的位置上。排列在前的网页由于表明有广告机会，对利润追逐者具有强烈的诱惑力，从而能为网页的拥有者创造经济利益。还有就是计算机病毒战争，令病毒制造者和杀毒软件的设计者们一争高下。这个动机是制造一般的破坏和拒绝服务，而非直接赚钱。

计算机网络安全是一场没有止境，且愈演愈烈的战役。保护者们强化网络、操作系统以及应用。而攻击者们在这三方面寻找薄弱环节。入侵侦察系统能搜寻出不同寻常的、可能是由黑客试探行为引起的活动模式。攻击者意识到这一点并且试图掩盖他们的踪迹，他们或许是采取间接工作，或许延长活动时间。或者相反，非常迅速地攻击。数据挖掘运用于这个问题上，试图去发现被入侵侦察系统忽略的、计算机网络数据中存在的入侵者踪迹之间的语义关系。这是一个大规模的问题：用来监测计算机网络安全的审核日志，即使是一个中等规模的组织每天的量也是要以十亿字节计的。

许多自动威胁侦察系统都建立在将当前的数据与已知的攻击种类的匹配上。美国联邦航空管理局开发了计算机辅助旅客预筛选系统（CAPPS），这套系统根据旅客航空记录对旅客进行筛选，并对那些需要额外行李检查的个人上做标记。例如，CAPPS是将现金支付归入高风险的一类，虽然明确的细节是不予公布的。但是，这个方法只能发现已知的或是能预计到的

威胁。研究者现在正在运用无指导的方案，例如进行异情和孤立点（outlier）侦察，试图检测出可疑的行为。除了侦察潜在威胁以外、异情侦察还可以用来侦察金融诈骗或者洗钱这样的非法活动。

数据挖掘现今正以国家防御的名义用在大量的数据中。各种不同种类的信息，例如金融交易、健康医疗记录、网络通讯等，正被挖掘出来建立各种走势图、社会网络模型以及监测恐怖分子的通讯联系。这些活动引起了人们对隐私问题的极大关注，促进了保护隐私
357 (privacy-preserving）的数据挖掘技术的发展。这些算法试图辨别存在数据中的模式却不直接进入原始数据，典型方法是使用随机值使其失真。为保护隐私，必须保证挖掘过程所获信息不足以重建原始数据。说来容易，要实现可就难了。

并非所有对抗性的数据挖掘都是针对穷凶极恶的活动的。在复杂的、有干扰的实时领域中的多代理(multiagent)系统包括了一些自治代理， 它们不仅要在一个团队中协作并且要与对手竞争。如果你无法用肉眼观察到这种情形，联想一下英式足球。机器人足球（Robo-soccer）是一个丰富而普及的领域，可用以探究机器学习是如何应用在如此困难的问题中的。球员不仅要磨炼基本功还要学习怎样相互配合以对付不同类型的对手。

最后，机器学习已被用来解决历史文献之谜，它揭示了一个试图隐藏身份的作者。如Koppel和Schler（2004年）所述，Ben Ish Chai是十九世纪末巴格达重要的希伯莱语学者。在他的大量文献中有两个文集，包括大约500封以希伯莱-亚拉姆文字写成的回答法律质询的信件。已经知道是他写了一个文集。虽说他声称在一个档案中发现了另一个文集，历史学家怀疑他也是另一个文集的作者，只是故意改变文风试图掩盖他的作者身分。这个案例给机器学习提出的难题是没有其他的文集可归属于这个神秘作者。虽然有一些已知的候选对象，但是这些信件是由其他任何一个人所写的可能性是相同的。一种称为“揭密”（unmasking）的新技术被开发出来了，利用它建立一种模型能够区别已知作者的作品A和未知作者的作品X，迭代去除那些对于辨别二者最为有用的属性，随着越来越多的属性被去除，交叉验证正确率不断下降，考察这个正确率的下降速度。假设前提是如果作品X是由作品A 的、那个想隐藏身份的作者所写，那么作品X和A之间的差别与作品X 和另外一个不同作者的作品B之间的差别相比较，差别会表现在相对较少的部分属性上。换句话说，将作品X与作品A、B分别比较时，随着属性的去除，与作品A相比较时的正确率曲线下降比与B相比较时快得多。B. Koppel和Schler 得出的结论是，Ben Ish Chai 确实撰写了神秘信件，他们的这项技术是一个令人注目的、新颖独创的、机器学习应用于对抗情形的例子。

8.5 无处不在的数据挖掘

本书的开始便指出了数据无处不在的事实。这些数据影响着普通人的生活，然而影响最
358 大的要数国际互联网了。目前，在网上有超过50亿份文件，总计大约20万亿字节 (TB)，而且还在呈指数级增长，大约每6个月翻一番。绝大多数美国消费者使用网络。而他们中无人能和信息激增同步。数据挖掘源于这个数据库所在的世界，文本挖掘使机器学习技术从公司移入家庭。无论何时，当我们被网络上的数据淹没时，文本挖掘担保我们有工具能驯服它（数据）。应用是众多的。寻找朋友并和他们联系、保持金融投资组合、在电子世界中讨价还价地购物、用于任何方面的数据侦察器，所有这些都可以是自动完成的，不需要程序设计。文本挖掘技术已经被运用来预测你要点击的下一个链接、为你整理文件、处理你的邮件。在这样一个数

据无处不在、同时又是杂乱无章的世界中，文本挖掘绝对是我们所需要的解决方法。

许多人相信互联网预示着一个更为强大的范例转变：无处不在的计算。随处可见小型的便携式装置，移动电话、个人电子助手、个人立体声录像播放器、数码相机、移动连接网络。有的设备已经综合了所有这些功能。它们知道我们的时空位置，帮助我们在社会空间通讯、组织个人计划、回顾我们的过去并将我们包容在世界信息空间里。在当今美国的任何一个中产阶级家庭里都可以轻松地找到许多处理器。它们互相之间并不交流，也不和全球信息系统交流，目前还没有。但它们终有一天会的，而这一天来到时，数据挖掘的潜力就会喷薄而出。

拿音乐制品来说。流行音乐是引导技术进步的先锋。索尼最初的随身听为今天随处可见的便携式电子设备铺平了道路，苹果公司的iPod率先开发了大容量的便携式存储。Napster的网络技术促进了对等协议（peer-to-peer protocols）的发展。类似Firefly的推荐系统将计算技术引入社会网络中。在不久的将来，能知晓内容的音乐服务将嵌入便携式设备。数据挖掘技术在网络化的音乐服务社区用户上的应用将是大量的，发现音乐潮流走向、追踪偏好和品位、分析收听习惯。

无处不在的计算将会把数字空间和现实世界的活动紧密地连接在一起。对许多人来说，臆测他们自己的计算机经历总是充满挫折感、神秘的技术、感到个人（能力）不足、乃至出现计算机机械故障，这看上去就像是场噩梦。然而，倡导者们指出情况并不会如此，如果是这样，便不可行了。当今的幻想家们预见到了一个“平静”的计算机世界，在这个世界里隐 359
蔽的机器在幕后默默地联合工作着，使人类的生活更加丰富和方便。它们处理的问题大到公司财务和家庭作业，小到一些令人烦恼的小事，诸如车钥匙在哪里、有停车位吗、上星期在Macy看到的那件T恤衫还在衣架上吗？当没有了电源时，时钟能知道正确的时间、微波炉能从因特网上下载新的菜谱、儿童玩具能够自动更新获得新游戏和新词汇。衣服标签能够跟踪洗涤、咖啡杯通知清洁工来清洗、如果没有人在房间里电灯开关将会节能、铅笔能将我们所画的进行数字化。在这个全新的世界里，数据挖掘在哪里呢？到处都是。

要指出现在尚不存在的未来的某个实例是困难的。然而，用户界面技术还有待提高。在直接操作的计算机界面上，许多重复性的任务，采用标准的应用工具不能实现自动化，迫使计算机用户必须重复进行相同的界面操作。这也是先前提到的挫折感的一个代表，由谁负责：我，还是机器？经验丰富的程序员会编写一些脚本程序由机器来完成这样的任务，但是随着操作系统在复杂层上的累计增加，程序员对机器施加命令的权力越来越小，并且当复杂的功能是被嵌入设备中而非通用计算机时，这个权力便消失了。

在示范编程（programming by demonstration）方面的研究使普通的计算机用户能够让机器自动完成任务而无需了解任何编程知识。用户只需知道执行这个任务的常规方法以便和计算机交流。一种系统称为Familiar，能帮助用户自动完成苹果机上的应用程序的重复执行任务。它不仅能完成这些任务而且还能执行它所未遇见过的新的任务。这是通过使用苹果的脚本语言从每个应用上收集信息并利用这些信息来做出预测而实现的。代理机还能容忍干扰。它告知用户它所做的预测，并结合考虑反馈信息。它具有适应性：为个体用户学习特定的任务。而且，它对每个用户的风格是敏感的。如果两个人都在传授一个任务，正好给出的也是相同的示范，Familiar系统不一定推断出相同的程序，它会根据用户的习惯进行调整，因为它是要从人机之间的交互历史中学习的。

Familiar应用标准的机器学习技术来推断用户的意图。用规则来评估预测，以使在每个阶

段都能提供给使用者最佳的预测。这些规则是有条件的，因此用户可以教导分类任务，诸如按文件种类进行文件整理，并根据文件的大小来赋予标签。它们是递增地学习的：代理通过记录人机之间的交互历史来适合个体使用者。

出现了许多困难。一是数据的缺乏。用户厌烦对一个任务重复示范几次，他们认为代理机应该理解他们正在做的事。一个数据挖掘者认为含100个实例的数据集是很小的，而用户示范一个任务几次便已恼火。困难之二是过多的属性。计算机环境拥有数以百计的属性，任何
360 行动都可能依赖这些属性。这意味着小的数据集极可能包含了一些看似极具预测力而实际却是无关的属性，还需要用特别的统计测试来比较假设。困难之三在于这种迭代的、不断改进的发展模式，这个特性导致数据挖掘应用失败。从理论上看，不可能为类似示范编程这样的交互式问题来建立一个固定的训练和测试集，因为代理机的每次改进，会通过模仿用户将如何反应来改变测试数据。困难之四是现有的应用程序只能提供有限地进入应用程序以及用户数据：成功操作所依赖的原始资料经常是被深埋在应用程序内部，却进入不了。

数据挖掘在工作中已被广泛运用。在我们阅读电子邮件和网上冲浪时，文本挖掘正将本书中的技术带入我们的生活中。将来，它可能和我们所能想象的不同。正在日益扩张的计算构架将为学习提供无法预言的机会。数据挖掘将在幕后扮演一个奠基的角色。

8.6 补充读物

关于大型数据集的文献极其可观，我们这里只能略举一二。Fayyad 和 Smith（1995年）论述了数据挖掘在庞大的科学实验数据中的应用。Shafer等人（1996年）描述了一种平行版本的自上而下的决策树归纳法。Mehta等人（1996年）为众多的磁盘数据集开发了一种有序的决策树算法。Breiman（1999年）叙述了如何能将任何算法应用于大型数据集，主要是通过将数据集分裂成较小的块，对结果进行装袋或提升。Frank等人（2002年）解释了相关修剪和选择方案。

虽然很重要，但有关将元数据纳入实际数据挖掘中的文献还是极少。Giraud-Carrier（1996年）考察了一个将领域知识编码成命题规则的方案，以及它在演绎和归纳上的应用。与归纳逻辑编程相关的、通过一阶逻辑规则来处理知识表达，则是由Bergadano和Gunetti（1996年）论述的。

文本挖掘是一个新兴领域，因而关于整个领域的综合论述还比较少：Witten（2004年）贡献了一个。Sebastiani（2002年）将大量的属性选择和机器学习技术运用于文本分类上。Martin（1995年）描述了文件聚类在信息检索方面的运用。Cavnar和Trenkle（1994年）演示
361 如何运用n-gram文件轮廓来正确地确定文件所使用的语言。支持向量机用于作者归属问题是由Diederich 等人（2003年）所叙述的。Dumais 等人（1998年）用相同的技术在大量训练文件的基础上，从受控词汇库里对文件赋予关键词汇。Turney（1999年）以及Frank等人(1999年)都研究了怎样使用机器学习从文件文本中提炼关键词汇。

Appelt（1999年）论述了许多关于信息萃取的问题。许多作者运用了机器学习来为模版的槽内填充内容提炼寻找规则，例如Soderland 等人（1995年）、Huffman（1996年）以及Freitag（2002年）。Califf 和Mooney（1999年）以及Nahm和Mooney（2000年）都探索了从张贴在互联网新闻上的招工广告中提炼信息的问题。Witten 等（1999年）报告了一种基于压缩技术、在连续的文本上寻找信息的方法。Mann（1993年）从美国国会图书馆所收到的文件中

发现了利比亚领导人卡扎非名字的多种表达形式。

Chakrabarti（2003年）写了一本优秀的、综合性的、关于网络挖掘技术的书籍。Kushmerick等人（1997年）发展了包装归纳技术。Tim Berners-Lee 等（2001年）介绍了语义网络，他早在十年之前便开发了国际互联网背后的技术。

第一篇关于垃圾邮件过滤的论文是由Sahami等人（1998年）写的。我们关于计算机网络安全的材料是源于Yurcik 等（2003年）。CAPPS系统的信息是来自从美国众议院关于飞行的文件（2002年），用于威胁侦察系统的无指导学习是由Bay 和Schwabacher (2003年) 论述的。当前的保护隐私数据挖掘技术所存在的问题是由Datta 等（2003年）确定的。Stone和Veloso (2000年) 从机器学习的角度来审视类似机器人足球运动的多代理系统。令人感兴趣的有关Ben Ish Chai的故事以及揭露他身份的技术来自Koppel和Schler（2004年）。

平静的计算世界景象和我们提到过的例子来自Weiser（1996年）以及Weiser和Brown（1997年）这两篇论述。关于示范编程的不同方法的更多信息可以从Cypher（1993年）和Lieberman（2001年）上找到更多的信息。Mitchell等人（1994年）报道了一些学习学徒的经历。Paynter（2000年）描述了Familiar。Good（1994年）所论述的排列测试是适合小样本问题的统计测试：Frank（2000年）讨论了它们在机器学习上的运用。 362

第二部分 Weka机器学习平台

第9章 Weka简介

经验表明，没有哪一种机器学习方案可以解决所有数据挖掘问题。放之四海而皆准的学习器不过是一种幻想。正如我们在本书中所强调的，真正的数据集是多种多样的。要想获取准确的模型，学习算法的偏好必须与该领域的结构相吻合。数据挖掘是一门实验性很强的学科。

Weka工作平台汇集了当今最前沿的机器学习算法及数据预处理工具。本书中所讨论的算法几乎都涵盖于其中。目的是为了让用户能够快速灵活地将现有的处理方法应用于新的数据集。它为数据挖掘实验的整个过程，包括准备要输入的数据，统计地评估学习方案，以及可视化输入数据及学习结果，提供了广泛的支持。Weka不但包含多样化的学习算法，还提供大量适应范围很广的预处理工具。用户可通过一个共用的界面操作运用所有已包含的工具组件，从而比较不同的学习算法，找出能够解决当前问题的最有效的方法。 365

Weka是由新西兰怀卡托大学开发的。Weka是怀卡托智能分析系统的缩写。在怀卡托大学以外的地方，Weka通常按谐音念成Mecca，是一种现今仅存活于新西兰岛的，具有好奇心的不会飞的鸟。Weka是用Java写成的，并且限制在GNU通用公众证书的条件下发布。它可运行于几乎所有的操作平台。在已经测试过的平台包括Linux，Windows和Macintosh操作系统，甚至还包括个人数字化助手。Weka提供了一个统一界面，可结合预处理及后处理方法，将许多不同的学习算法应用于任何所给的数据集，并评估由不同的学习方案所得出的结果。

9.1 Weka中包含了什么

Weka通过实现各种学习算法，使用户能够很容易地将其应用于所要处理的数据集中。Weka还包含了种类繁多的用于数据集转换的工具，比如在第7章中讨论过的用于离散的算法。用户可以先将一个数据集进行预处理，然后置其于一种学习方案中，并对所得出的分类器及其性能表现做出分析。整个过程无须用户编写任何程序代码。

Weka工作平台包含能处理所有的标准数据挖掘问题的方法：回归、分类、聚类、关联规则挖掘以及属性选择。对所要处理的数据进行分析是整个工作必不可少的一环，Weka提供了许多用于数据可视化及预处理的工具。所有算法对所要输入的数据都有以下要求：其数据形式必须是在第2.4节中描述过的ARFF格式，并要求以单一关系列表的形式输入。这些数据可从文件中读取或由数据库查询所产生。

Weka的使用方式之一是将一种学习方法应用于一个数据集，然后分析其输出，从而更多地了解这些数据。另外一种方式则是使用已学习到的模型对新的实例做出预测。第三种方式是应用几种不同的学习器，再根据它们的性能表现选择其中一种用来做预测。学习方法又称作分类器，用户可在Weka互动式界面的菜单中选择一种想要的分类器。许多分类器带有可调

节的参数，这些参数可通过属性列表或对象编辑器进行更改。所有分类器的性能都是通过一个共同的评估模块进行衡量。

Weka平台中最有价值的部分是真实学习方案的实现。其次当属数据预处理工具，也称作过滤器。与选择分类器一样，用户也是从菜单中选择能满足其要求的过滤器。我们以后会谈到如何使用不同的过滤器，并列出过滤器所用的算法，及描述不同过滤器的参数。Weka中还包含了一些算法的实现，如学习关联规则，未指定类值的聚类数据，以及从数据中选择相关
366 属性等。我们会简单加以叙述。

9.2 如何使用Weka

使用Weka的最简单方法是通过称作探索者（Explorer）的图形用户界面。通过这个用户界面，所有Weka的功能都可以由菜单选择及表单填写的方式完成。例如，用户可从ARFF文件（或电子数据表）中快速读取一个数据集，并根据此数据集建立决策树。然而建立决策树仅仅是开始，还有很多其他算法有待于探索。探索者用户界面在这里大显身手。它通过将选项转化为菜单，将不适用的选项设定为不可选，以及将用户选项设计成表格填写的形式引导用户一步一步按照合适的顺序完成对算法的探索。Weka还对所含工具给出了用法提示，即当鼠标光标在屏幕上移至相应工具上时，以弹出窗口的形式解释该工具如何使用，这对于用户使用工具极有帮助。合理的默认值设置使得用户能以最小的工作量取得预期的结果。当然，用户必须对所做的操作进行思索，才能理解所得结果的确切含义。

Weka还包含了另外两个图形用户界面。*知识流*（Knowledge Flow）界面使用户能够自己设置如何处理流动中的数据。探索者用户界面的一个根本缺陷在于它将所需的数据全部置于主存中，一旦用户打开一个数据集，所有数据会被全部读取进来。这意味着此种处理方式只适用于小至中等规模的问题。Weka还包含一些能够处理非常大型的数据集的递增算法。知识流界面允许用户在屏幕上任意拖动代表学习算法和数据源的方框，并将它们结合在一起进行设置。这样使得用户能够通过将代表数据源、预处理工具、学习算法、评估手段以及可视化模块的各个部件组合在一起，形成一个数据流。如果用户所选的过滤器和学习算法具有递增学习功能，大型数据集的递增分批读取及处理即可实现。

Weka的第三个界面*实验者*（Experimenter）是专门设计来帮助用户解答实际应用中所遇到的一个基本问题，即在将分类及回归技术运用于实践时，对于一个已知的问题，哪些方法及参数值能够取得最佳效果？仅仅靠推测是无法回答这个问题的。我们设计这样一个工作平台的原因之一就是提供一个工作环境使得用户能够将不同的学习技术进行比较。虽然通过探索者界面也可互动式地做到这一点，然而使用实验者界面，用户可令处理过程自动化，因为它能使具有不同参数设定的分类器和过滤器在运行一组数据集时更加容易，收集性能统计数据及实现显著性测试时更加简便。高级用户还可利用Java远程方法调用（RMI）方式在实验者界面上将计算负荷分布到多个机器上。这样，用户即可设定好大规模统计实验，然后令机器自动运行。

隐藏在这些互动式界面背后的是Weka的最基本功能。这些功能涵盖了Weka系统的全部特
367 色并可通过键入文本命令的原始方式来实现。当用户打开并运行Weka后，必须从如下四种不同的用户界面中做出选择：探索者，知识流，实验者和命令行界面。我们会在下面依次介绍这些界面。大多数用户都会选择探索者界面，至少在首次使用Weka时如此。

9.3 Weka的其他应用

使用Weka时，一项重要的可利用资源是在线帮助文件。这个文件由源代码自动生成，准确地反映了源代码的结构。我们会解释如何使用这个文件，给出Weka的主要框架，并对含有指导性学习方法的部分，数据预处理工具的部分，以及其他学习方案的部分加以强调。Weka是一个处在不断发展中的系统，在线文件总是在不断更新。因此，这份由源代码自动生成的文件可确保是最新版本，而且是产生完整可用的算法列表的唯一途径。进一步说，这个文件对于那些想要提升到较高层次并在他们自己的Java程序中访问Weka程序库，或者希望编写并测试他们自己的学习方案的用户来说也是极为必要的。

在大多数数据挖掘程序中，机器学习只是大型软件系统中的一小部分。如果用户要编写一个自己的数据挖掘软件，最好在其软件的代码中调用Weka的程序。这样做用户只需额外编写最少量的代码，即可解决他们软件中有关机器学习的子问题。我们以一个用Java写成的有关数据挖掘的简单程序作为范例，来说明如何调用Weka中的程序。这样用户即可对Weka中代表实例，分类器和过滤器的基本数据结构加以熟悉。

如果用户要成为机器学习算法的专家（或者用户已经是专家），可能他们会实现自己的算法，无需把注意力放在一些索然乏味的细节上，例如从文件中读取数据，实现过滤算法，或编写代码来评估所得结果等等。如果用户真想这么做的话，以下应该是个好消息：Weka中已经包含所有以上这些功能。要充分利用Weka的这些功能，用户必须透彻地了解Weka的基本数据结构。为了帮助用户做到这一点，我们会进一步探讨这些结构并且利用一个示范性的分类器的实现过程加以解释。

9.4 如何得到Weka

Weka可由以下网址获取：http://www.cs.waikato.ac.nz/ml/weka。用户既可以下载一个与具体操作系统相匹配的安装文件，也可以下载一个可执行的Java包文件（jar file），然后在已安装了Java的机器上以通常的方式运行。我们建议用户现在就下载并安装，然后按照后面章节中给出的范例进行操作。 368

第10章 Explorer界面

通过Weka的主要图形用户界面Explorer（探索者），Weka的所有功能皆可以菜单选择或表单填写的方式来访问到。图10-1展示了探索者界面。在探索者界面顶部的六个不同的标签，表示六个不同的面板，分别对应着Weka所支持的不同的数据挖掘方式。

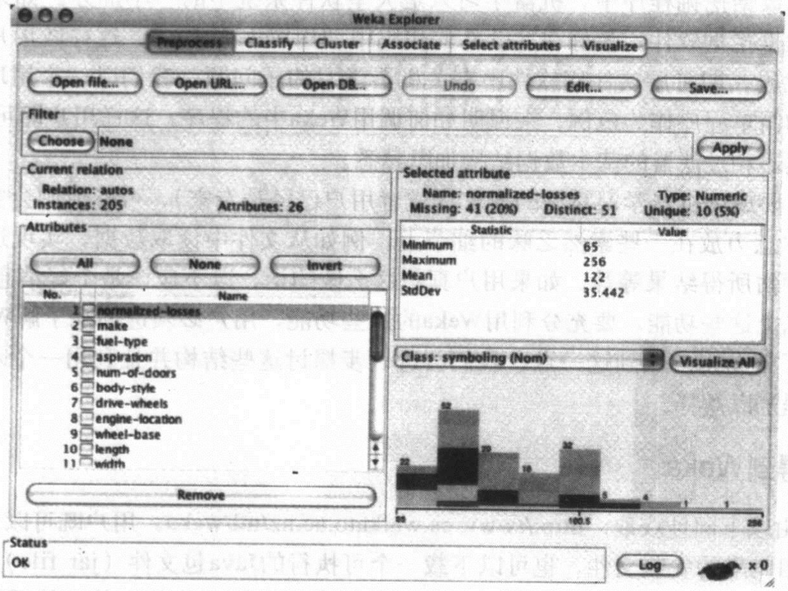


图10-1 探索者用户界面

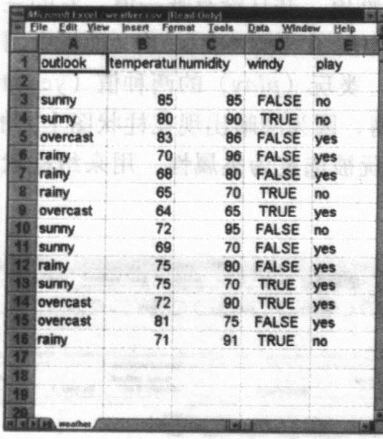
10.1 开始着手

假设用户有一些数据并且想用这些数据建立一个决策树。首先，用户需要准备数据，启动探索者界面，并载入数据。其次，选择一种构建决策树的方法将树建立起来，并解析得出的结果。使用不同的构建决策树的算法或不同的评估手段来重复以上过程并不难。这体现在用户可在探索者界面中任意选择每次过程所得出的不同结果，评估基于不同数据集所建立的模型，并且图形化所得出的模型及数据集本身，包括使用该模型所产生的任何分类误差。

10.1.1 准备数据

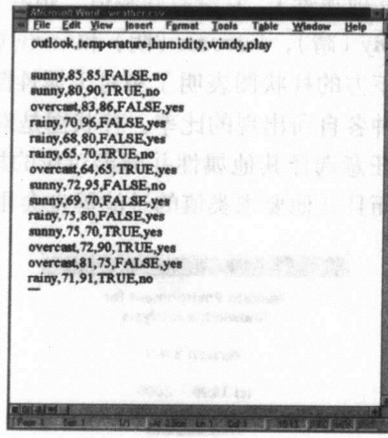
数据通常存储于电子数据表或数据库中。然而，Weka存储数据的原始方式是ARFF格式（第2.4节）。由电子数据表转换为ARFF格式非常容易。ARFF文件是由一组实例组成，并且每个实例的属性值由逗号分开（图2-2）。大多数电子数据表及数据库程序允许用户将数据导入逗号分隔数值（CSV）格式的文件中，该格式文件是由一组记录组成，且每项记录中的每个条目由逗号分隔。做完这一步后，用户只需将文件载入一个文本编辑器或文字处理软件，用

@relation标签给该数据集取个名字，用@attribute标签加入属性信息，再另起一行键入@data，最后将文件以纯文本方式存储即可。如图10-2中所示是一个取自1.2节中的天气数据的Excel电子数据表，载入微软Word处理器后以CSV格式呈现的同一组数据，以及以手动方式转换成ARFF文件后的结果。其实，用户自己无需真正履行这些步骤来建立ARFF文件，正如后面将要描述的一样，探索者能够直接读取CSV电子数据表。



	A	B	C	D	E
1	outlook	temperatu	humidity	windy	play
2					
3	sunny	85	85	FALSE	no
4	sunny	80	90	TRUE	no
5	overcast	83	86	FALSE	yes
6	rainy	70	96	FALSE	yes
7	rainy	68	80	FALSE	yes
8	rainy	65	70	TRUE	no
9	overcast	64	65	TRUE	yes
10	sunny	72	95	FALSE	no
11	sunny	69	70	FALSE	yes
12	rainy	75	80	FALSE	yes
13	sunny	75	70	TRUE	yes
14	overcast	72	90	TRUE	yes
15	overcast	81	75	FALSE	yes
16	rainy	71	91	TRUE	no
17					
18					
19					
20					

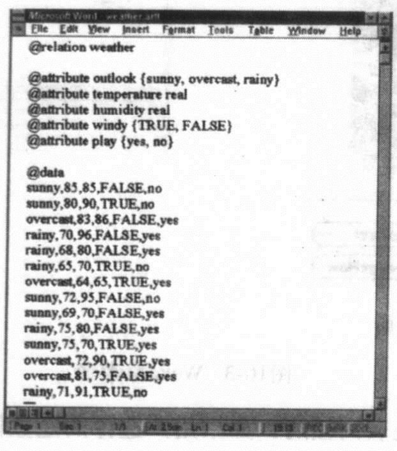
a) 电子数据表



outlook,temperature,humidity,windy,play

sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no

b) CSV格式



@relation weather

@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no

c) ARFF格式

图10-2 天气数据

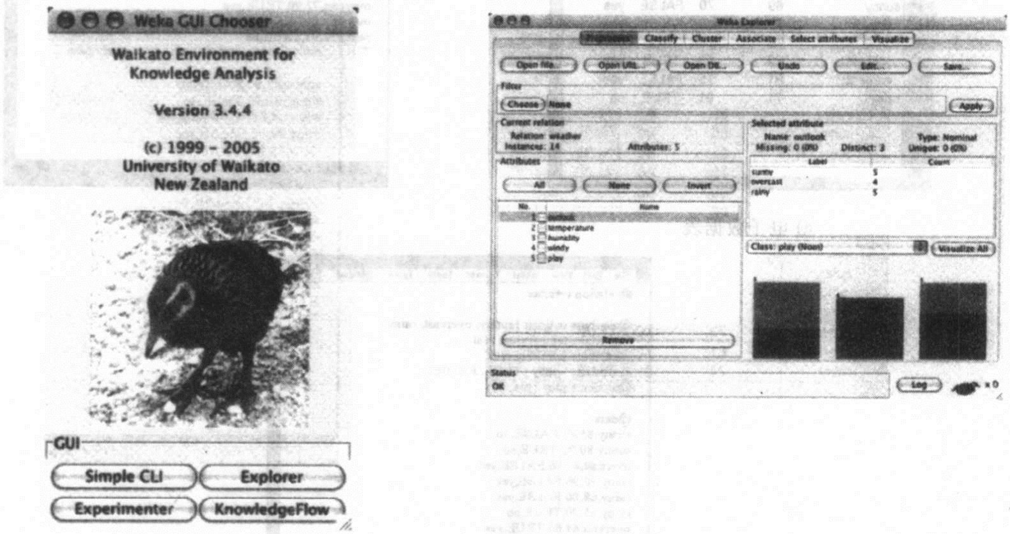
10.1.2 将数据载入探索者

现在将数据载入探索者并开始分析。启动Weka进入图10-3a所示画面。从画面底部的四个图形用户界面选项中选择探索者（*Explorer*）。（其他选项在前面的章节中已做过介绍：*SimpleCLI*是旧式的命令行界面。）

接下来用户看到的就是探索者的主面板图10-3b。实际上，图10-3b所示的是用户已经将天气数据载入探索者后的画面。面板顶部的六个标签代表着探索者所支持的基本操作功能。现在进入了预处理阶段（*Preprocess*）。单击*Open file*（打开文件）按钮，通过弹出的标准对话框 369~371

用户可选择文件。选择名为*weather.arff* 的文件。如果该文件是CSV格式的，则将其从ARFF数据文件改为CSV数据文件。当用户指定一个*.csv*文件时，该文件会自动转换成ARFF格式。

文件载入后，画面应如图10-3b所示。由面板中所载入的数据集可知：它含有14个实例和5项属性（面板中间偏左）；这5项属性是阴晴（*outlook*），温度（*temperature*），湿度（*humidity*），刮风（*windy*）和玩（面板左下方）。第一项属性阴晴是默认选项（用户可通过单击选择其他属性），它没有残缺值，但有三个不同的属性值，并且没有唯一值。它的三个属性值是*sunny*（晴），*overcast*（阴）和*rainy*（雨），且分别出现了5次、4次和5次（面板中间偏右）。面板右下方的柱状图表明了相对于属性阴晴的每个值，类玩（*play*）的两种值（yes和no）中的每一种各自所出现的比率。柱状图是对应于属性阴晴，因为阴晴出现在柱状图上方的框中，用户可任意选择其他属性并画出对应的柱状图。这里玩被选定为类属性；用来给柱状图加上色彩，而且其他要求类值的过滤器也会用到它。



a) 选择探索者界面　　　　b) 读入天气数据

图10-3　Weka探索者

图10-3b中的属性阴晴是名词性的。如果用户选择的是数值性属性，框中会出现最小值、
372 最大值、平均值和标准偏差。在这种情况下，柱状图所显示的是类作为该项属性的一个函数的分布。（图10-9给出了该种情况下的一个例子。）

用户可通过单击复选框和*Remove*（删除）按钮来删除属性。单击*All*（全部）则选中全部属性，*None*（无）表示不选，*Invert*（反转）则反向转换目前的选择。用户可通过点击*Undo*按钮撤消所做的改动。点击*Edit*按钮会弹出一个编辑器。通过该编辑器，用户可检查数据，搜索具体的值并对其进行编辑，以及删除实例和属性。在每个值以及每列的表头上右键点击会弹出相对应的上下文菜单。

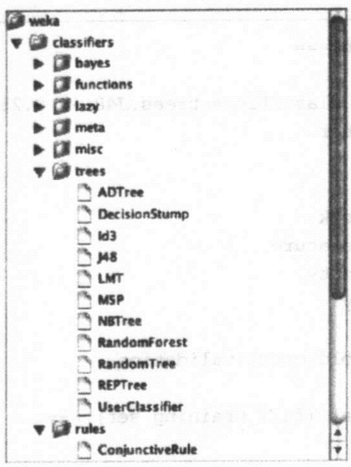
10.1.3　建立决策树

要想了解第6.1节中描述过的C4.5决策树学习器是如何处理数据集的，我们必须利用J4.8算法，这个算法是C4.5学习器的Weka实现。（J4.8算法实际上实现了一个被称作C4.5的修正版

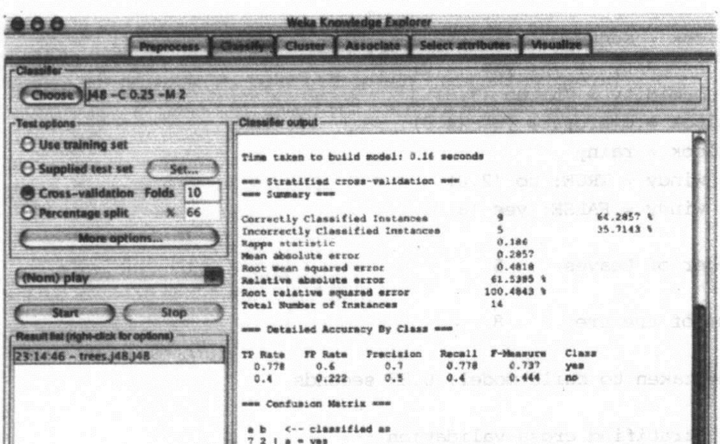
8的较新的且略有改进的版本，这也是在它的商业版C5.0推出前，该算法家族中的最后一个公开的版本。）单击*classify*标签，得到如图10-4b所示画面。该画面所显示的实际上是用户对天气数据做出分析后得到的结果。

第一步选取分类器。单击左上方的*Choose*按钮，在随后打开的如图10-4a所示的层级式菜单中的*trees*部分找到*J48*。该菜单的结构代表着Weka源代码的模块式架构，这部分我们会在第13章中加以描述。用户现在所要做的只是打开相应的层级（即箭头），待选分类器总是出现在层级的最底层。一旦被选中，*J48*以及它的相关默认参数值就会出现在*Choose*按钮旁边的条形框中，如图10-4b所示。单击这个条形框会打开J4.8分类器的对象编辑器，编辑器会显示J48的各个参数的含义，且用户可根据需要更改参数的值。探索者通常会合理地设定这些参数的默认值。

选定分类器后，单击*Start*按钮使其开始工作。Weka每次运行的时间较短。在它工作时，坐在图10-4b右下角的小鸟会即时起舞，随后就会产生如图10-4b所示主面板上的结果。



a) 在分类器列表中找到它



b) *Classify*标签页

图10-4　使用J4.8

10.1.4 查看结果

图10-5列出了全部输出结果（图10-4b所示只是后半部分）。在该结果的开头给出了数据集概要并注明所用的评估方法是10折交叉验证。该方法是默认的，如果用户仔细观察图10-4b会发现左侧的交叉验证（*Cross-validation*）一项是选中的。再往下是一个修剪过的决策树的文
373~374 档表格。首先根据属性阴晴进行分割，然后分别根据湿度和刮风在第二层次上进行分割。在这个树结构中，冒号后面的是分配到某个具体叶的类标志，类标志的后面是到达该叶的实例的数量。该数量是以十进制数表示的，这是因为所用的算法在处理残缺值时，使用按比例分解过的实例。如果有被错误分类的实例（图中没有这样的实例），这种实例的数量也会列出。比方说，2.0/1.0表示共有两个实例到达了该叶，其中一个是被错误分类的。在树结构的下方给出了该树所含的叶的数量，然后是节点的总数（树的大小）。有一种方式可以更加图形化地浏览树的结构，见10.1节。

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     weather
Instances:    14
Attributes:   5
              outlook
              temperature
              humidity
              windy
              play
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------

outlook = sunny
|   humidity <= 75: yes (2.0)
|   humidity > 75: no (3.0)
outlook = overcast: yes (4.0)
outlook = rainy
|   windy = TRUE: no (2.0)
|   windy = FALSE: yes (3.0)

Number of Leaves  :   5

Size of the tree :    8

Time taken to build model: 0.27 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances           9               64.2857 %
```

图10-5 J4.8决策树学习器的输出结果

```
Incorrectly Classified Instances         5               35.7143 %
Kappa statistic                          0.186
Mean absolute error                      0.2857
Root mean squared error                  0.4818
Relative absolute error                 60      %
Root relative squared error             97.6586 %
Total Number of Instances               14
=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
  0.778     0.6       0.7         0.778    0.737    yes
  0.4       0.222     0.5         0.4      0.444    no

=== Confusion Matrix ===

 a b   <-- classified as
 7 2 | a = yes
 3 2 | b = no
```

图 10-5（续）

接下来是对该决策树的预测能力所做的评估。对图中所举的例子来说，该评估是通过默认的分层10折交叉验证得到的，如图10-4b所示。由图中可知，超过30%的实例（14个中的5个）在交叉验证中被错误分类。这表明，相比较根据同样数据源得来的独立测试集所可能得到的结果来说，由训练数据所得的结果是过于乐观了。用户还可在（5.7节所述）最下方的混淆矩阵中看到，有两个属于yes的实例被分入类no，同时类no的3个实例被分给了类yes。

除了列出分类误差，评估模块还在输出结果中给出Kappa统计数据（5.7节）和平均绝对误差，以及由决策树指定的类的概率估计的根均方差。根均方差是平均二次平方损失的平方根（5.6节）。平均绝对误差也是以类似的方法计算出来的，只是将平方差换成绝对差。评估模块还输出基于先验概率基础上的相对误差（即，由后面将要描述的*ZeroR*学习方案所得到的 375~376
概率）。最后，它还会针对每个类给出5.7节中描述过的一些统计数据。

10.1.5 重做一遍

用户可以很容易地用不同的评估手段将J4.8再运行一遍。选中*Use training set*（图10-4b左上方），然后单击*Start*按钮。很快，分类器输出结果给出了对训练集所导出的模型的表现情况而不是交叉验证的结果。此评估有些过于乐观了（5.1节）。然而，这类评估通常代表了所用模型对初次使用的数据处理能力的上限，因此依然不无帮助。在这种情况下，全部14个训练实例皆分类正确。有些时候，某个实例也许会决定对一些实例不作分类。在这样的情况下，这些实例作为未分类实例（Unclassified Instances）列出。对Weka中的大多数学习方案来说，这样的情况通常不会出现。

图10-4b所示面板上的其他测试选项是：*Supplied test set*（提供的测试集），即用户指定一个含有测试集的文件；*Percentage split*（百分比分割），这种情况下，用户可将一定比例的数据留作测试用。单击*More options*（更多选项）按钮，并检视相应条目，用户可在屏幕上输出每个实例的预测精度。面板上还有其他一些选项，如省略一些输出项目，设定输出结果中含

有像熵评估度量和成本敏感评估一类统计数据等功能。对于后者，用户必须在*Classes*框中键入类的数量（然后敲，*Enter*或*Return*键终止），在随后得到的成本矩阵（5.7节）中按要求修改相关数据。

在图10-4b左下方的小窗口中有一条点亮的横向的栏，这里给出的是所得结果的历史列表。每次用户运行一个分类器，探索者会加入一条新栏。因为到目前为止该分类器被运行过两次，该列表中有两条横栏。要返回到前面的结果集，只需单击相关的横栏，对应此次运行的输出就会出现在分类器输出窗口中。该功能使得浏览不同的分类器或评估方案并反复比较他们的结果变得非常容易。

10.1.6 运用模型

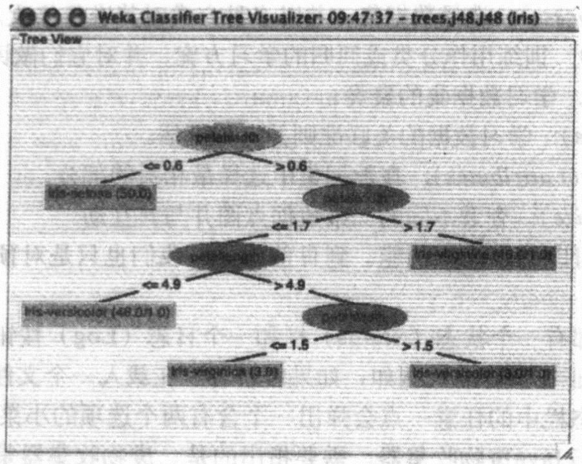
输出结果历史列表是使用探索者某些强大功能的入口。当用户右键单击其中一项时，随后出现的菜单允许用户在单独的窗口查看结果，或将结果存储下来。更重要的是，用户可将Weka所产生的模型以Java对象文件的形式保存起来。用户还可重新载入以前保存过的模型，这样会在历史列表中产生一个新的记录。如果现在用户提供一个测试集，则可用这个新测试集重新评估以前的旧模型。

右键单击弹出的菜单中几个选项允许用户以不同的方式可视化所得各种方法的结果。在
377 探索者界面的顶部有一个*Visualize*标签，但这个标签不同的地方在于：它是用来显示数据集而不是某个模型的输出结果的。通过右键单击历史列表中的项目，用户可看到分类器的误差。如果所用模型是树或贝叶斯网络，用户可看到它的结构。用户还可浏览边差曲线（7.5节）及不同成本和阈曲线（5.7节）。对于成本和阈曲线，用户可从分菜单中选取一个类值。然后利用菜单中的*Visualize threshold curve*选项来查看改变不同的、使某个实例分入该类值的概率阈值后所得曲线的效果。可供用户选择的曲线范围很广，包括ROC和反馈率-精确率曲线（表5-7）。要选取曲线，从所给菜单中设定合适的X和Y坐标轴。例如，对一个ROC曲线来说，将X设为*False positive rate*，Y设为*True positive rate*，或X设为反馈率（*Recall*），Y设为精确率（*Precision*），则得到反馈率-精确率曲线。

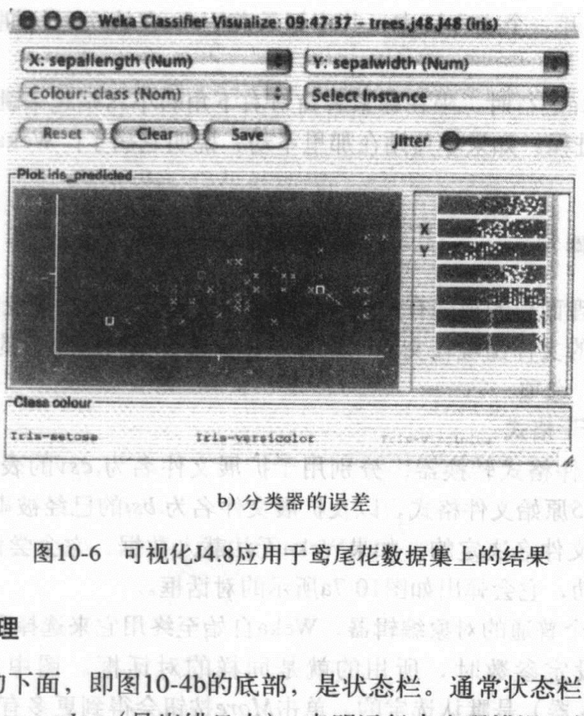
图10-6给出了两种方式查看利用J4.8对鸢尾花数据集进行分类的结果（1.2节），这里没有使用天气数据集是因为用鸢尾花数据集生成的图更有趣。图10-6a显示了所产生的树。在窗口的空白处右键单击弹出一个菜单，用户可设定图形尺寸自动调整或者说使树更适合窗口的大小。拖动鼠标可使树在窗口中任意移动。如果树已经根据所用算法存储下来，还可将任意节点上的实例数据图形化。

图10-6b用二维点阵显示了分类器误差。用户可利用界面顶部的选择框为坐标X和Y选择属性。还有一个办法是单击位于二维点阵右侧的带斑点的水平条中的任何一条：左键单击确定X坐标，右键确定Y。每个水平条显示的是对应着该属性的实例分布。字母X和Y会分别出现在用户所选定的作为相应X和Y坐标的水平条旁边。

代表数据的点用不同的颜色区分各自所属的类：蓝，红和绿，分别代表鸢尾花Setosa，Versicolor和Virginica（见屏幕底部标注）。被正确进行分类的实例用十字交叉表示，未被正确分类的实例则用小方框标明（图10-6b中共有3个小方框）。用户可单击任何一个代表实例的点查看其详细信息：实例的编号，属性值，实际所属的类和预测出的类。



a) 树



b) 分类器的误差

图10-6 可视化J4.8应用于鸢尾花数据集上的结果

10.1.7 运行错误的处理

在历史结果列表的下面，即图10-4b的底部，是状态栏。通常状态栏只是简单显示OK。有时候状态栏会显示*See error log*（见出错日志），表明运行中出现错误。例如，各种不同的需由用户在面板上进行设定的选项通常都有一些限定条件。绝大多数情况下，界面上不兼容的选项会被设成不可选（灰色），即用户无法选择它们。但偶尔也有这样的情况，太复杂的人机互动导致用户能够选择一些不兼容的选项。在这种情况下，当Weka意识到出错时，通常是用户单击*Start*时，状态栏会出现提示。要查看出错信息，单击界面Weka鸟的右下角的左侧*Log*按钮。 378~379

10.2 探索"探索者"

在图10-3b和图10-4b中我们简单介绍了探索者窗口顶部六个标签中的两个。在此，对他们的功能做一下小结:

1) 预处理（*Preprocess*）：选择数据集，并以多种方式对其进行修改。

2) 分类（*Classify*）：训练用作分类或回归的学习方案，并对它们做评估。

3) 聚类（*Cluster*）：学习数据集的聚类。

4) 关联（*Associate*）：学习数据的关联准则并对其评估。

5) 选择属性（*Select attributes*）：在数据集中选择最相关的部分。

6) 可视化（*Visualize*）：查看不同的二维数据点图并与其互动。

在每个标签页都可以使用全部功能。到目前为止，我们也只是对预处理和分类面板做了肤浅的探讨。

在每个面板的底部有一个状态（Status）栏和一个日志（Log）按钮。用户可由状态栏中显示的信息知道Weka正在做什么。例如，如果探索者正在载入一个文件，状态栏中会显示这个信息。右键单击状态栏中的任意一点会弹出一个含有两个选项的小菜单：显示Weka目前可用的存储器容量和运行Java废物收集器。需要指出的是，废物收集器是作为一个后台任务不间断运行的。

单击Log按钮会打开一个文本日志，其中记录着Weka在该次运行期间所进行的所有活动以及各项活动的时间。

如前所述，当Weka运行时，坐在探索者窗口右下角的小鸟站起来跳舞。×旁边的数字显示有多少同时进行的进程。如果小鸟站在那里不动，说明它病了！Weka出现运行错误，用户必须重新启动探索者。

10.2.1 载入及过滤文件

在图10-3b中预处理面板的顶部有一些按钮是用来打开文件，站点及数据库的。初始状态，只有名字以*.arff* 结尾的文件出现在文件浏览器中，要想浏览其他格式的文件，调整文件选择框中的*Format*（格式）选项。

将文件转换为ARFF格式

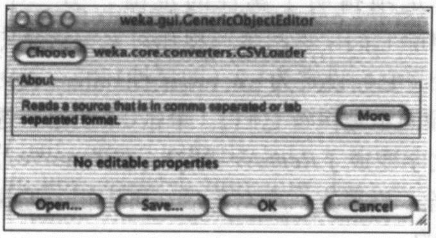
Weka带有三种文件格式转换器：分别用于扩展文件名为*.csv*的表格文件，扩展文件名为*.names*和*.data*的C4.5原始文件格式，以及扩展文件名为*.bsi*的已经被串行化的实例。可适用的转换器是根据扩展文件名决定的。如果Weka无法载入数据，它会尝试着将其当作ARFF格式来处理。如果不成功，它会弹出如图10-7a所示的对话框。 380

图10-7a所示是一个普通的对象编辑器，Weka自始至终用它来选择和设定对象。例如，当用户为一个分类器设定参数时，所用的就是同样的对话框。图中用于载入*.csv*文件的*CSVLoader*（CSV载入器）是默认选定的，单击*More*按钮会得到更多有关CSVLoader的信息，见图10-7b。阅读帮助文件总是有益的！就此例来说，帮助文件说明电子数据表的第一行决定了属性的名字等。单击OK即可应用这个转换器。图10-7c中，单击*Choose*可从列表中选择不同的转换器。

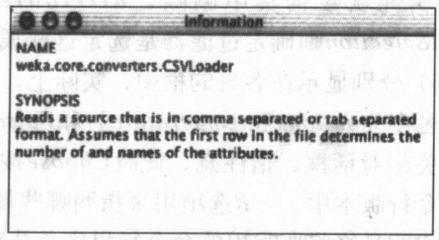
ArffLoader（Arff载入器）是第一个选项，我们之所以面临这些选项是因为之前的步骤不成功。*CSVLoader*是默认的，要更换其他选项，单击*Choose*。第三个选项是C4.5格式，内有两个文件构成一个数据集，其中一个文件是域名，另一个是实际数据。第四个选项串行化的实例（serialized instances）用来重新载入已经被存储为串行化的Java对象的数据集。任何Java对象皆可以该种形式存储并重新载入。作为一种原始的Java格式，它比载入ARFF格式文件要

快得多。ARFF格式文件需经过解析和检查。当需要不停地重新载入大的数据集时，将它们存成这种形式会很有用。

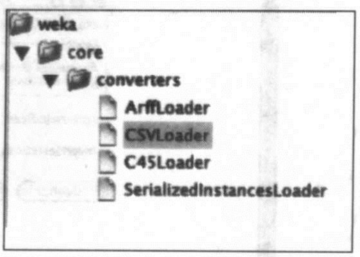
图10-7a所示的通用对象编辑器的其他功能包括：存储（*Save*），将一个经过设置的对象存储起来，和打开（*Open*），打开一个以前存储过的对象。这些功能对于图中所举的例子也许用不上，但是这个编辑器的其他一些面板有很多可编辑的属性，当用户设定这些属性遇到麻烦时，可将那些已设定过的对象存储起来以备将来使用。



a) 编辑器



b) 更多信息（单击*More*）



c) 选择转换器（单击*Choose*）

图10-7 通用对象编辑器 381

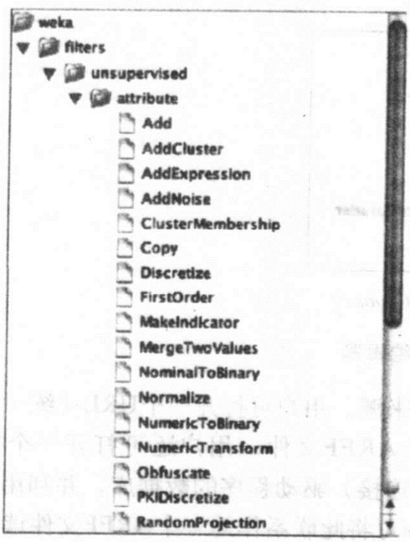
用户计算机上的文件并不是Weka数据集的唯一来源。用户可打开一个URL（统一资源定位器），Weka可利用HTTP协议由Web网下载一个ARFF文件。用户还可打开一个数据库（*OpenDB*），可以是任何已装载JDBC（Java数据库连接）驱动程序的数据库，并利用SQL选择语句获取实例。选择语句会返回一个联系，Weka可将此联系作为一个ARFF文件读入。欲使该功能也能用于用户的数据库，用户必须对Weka发行配套包中的*Weka/experiment/DatabaseUtils.props*文件进行修改。修改的方式是将用户数据库的驱动程序添加到这个文件中。（读取这个文件的方式是展开Weka发行配套包中的*Weka.jar*文件。）

利用图10-3b的Save按钮，数据可存储为前述格式中的任意一种。除了载入和存储数据集功能，用户还可通过预处理面板，过滤数据集。过滤器是Weka的一个重要组件。

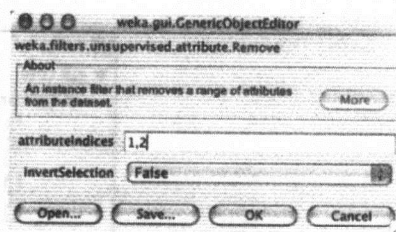
使用过滤器

单击图10-3b中的*Choose*按钮（左上方）得到图10-8a所示的一列过滤器。实际上，用户得到的是一个折叠起来的版本，单击其中的箭头可展开它的内容。我们会描述如何利用一个简单的过滤器从一个数据集中删除指定的属性，换句话说，进行手动属性选择。通过在复选框中打勾，然后单击*Remove*（删除）按钮来选择相关属性的方式可以更加容易地取得同样效果。无论以何种方式，我们会用一个范例明白无误地描述同样的过滤器操作过程。

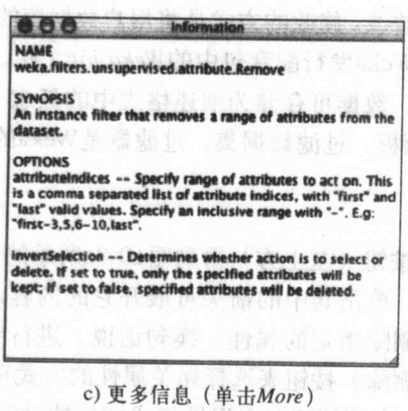
*Remove*是一个无指导属性过滤器，需要进一步向下拉动页面才能看到它。如果被选中，它就会出现在*Choose*按钮的旁边，与其同时出现的还有它的参数值，就图中的例子来说，横栏中只是简单地显示“Remove”。单击该过滤器横栏会弹出一个通用对象编辑器，通过该编辑器用户可检查并修改过滤器的属性。（用户在前面的章节中进行过类似的操作，即在图10-4b中单击*J48*横栏，打开J4.8分类器的对象编辑器。）Remove过滤器的对象编辑器如图10-8b所示。学习如何使用该编辑器，单击*More*显示如图10-8c所示信息。由图中可知，过滤器将部分属性从数据集中删除。*attributeIndices*选项指明了属性的范围，另一个选项*invertSelection*则确定过滤器是选定这些属性还是删除它们。这两个选项在对象编辑器中（图10-8b）分别显示在各自的框中，实际上，我们已经将它们设定为1,2（即属性1和2，相对应的属性名字为*outlook*和*temperature*）和*False*（即删除而不是保留它们）。单击OK使这些选项生效并关闭对话框。请注意，此时*Choose*按钮旁的文字变成了*Remove-R1,2*。在Remove过滤器的命令行版本中，－*R*选项用来指明哪些属性被删除。设置完一个对象后，再审视一遍探索者所设定的最终将要得出的命令行程序总是有益无害的。



a) 过滤器菜单



b) 对象编辑器



c) 更多信息（单击*More*）

图10-8　选择过滤器

单击*Apply*（图10-3b右侧）即可应用该过滤器。随即出现屏幕如图10-9所示。该图与图10-3b的唯一区别是图10-9只有三个属性，湿度，刮风和玩。此时靠近屏幕上方的一排按钮中的第四个按钮变为可选。*Undo*会撤销过滤器的动作并恢复原来的数据集。当用户用不同的过滤器做实验时，这个功能会很有用。

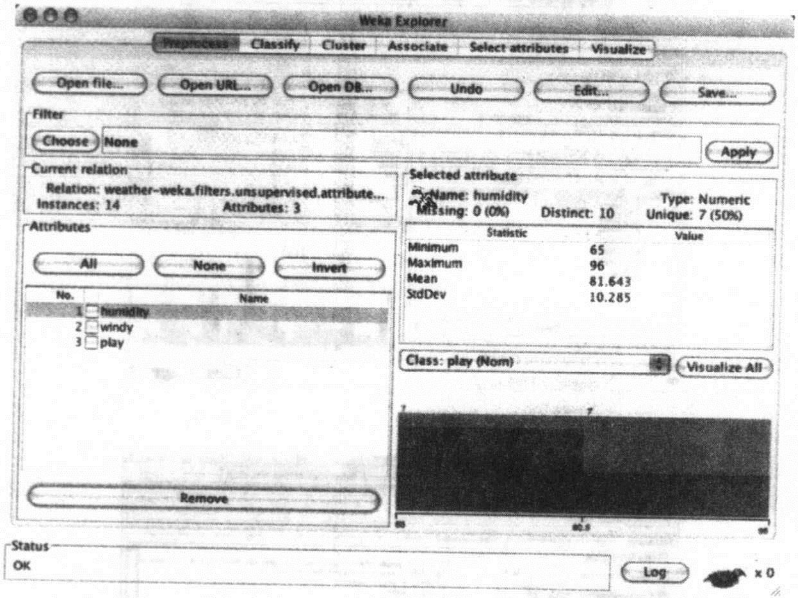


图10-9 删除两个属性后的天气数据

第一个属性湿度是选中的，并在面板右侧对它的值做了总结。作为一个数值性属性，这里显示的是它的最小值，最大值，平均值和标准偏差。在这些值的下面是一个柱状图，它显示的是属性玩的分布情况。不幸的是这个柱状图简单的可怜，因为这个属性的值的变化过于单调以至于所显示的只是两个同等宽度的方形。现实当中的数据集会产生内容更丰富的柱状图。

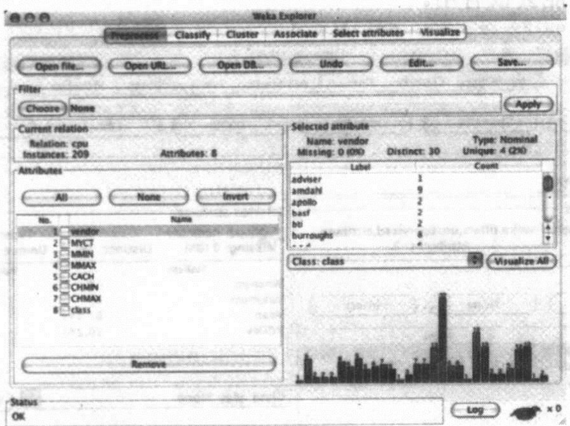
10.2.2 训练和测试学习方案

用户可通过分类面板来训练和测试用于分类或回归的学习方案。第10.1节讲解了如何解读一个决策树学习器的输出结果，给出了由评估模块自动产生的性能指数。这样的解读对所有用来预测属于某一范畴的类的模型都是一样的。其次，在评估数值预测的模型时，Weka产生的是一组不同的性能衡量标准。

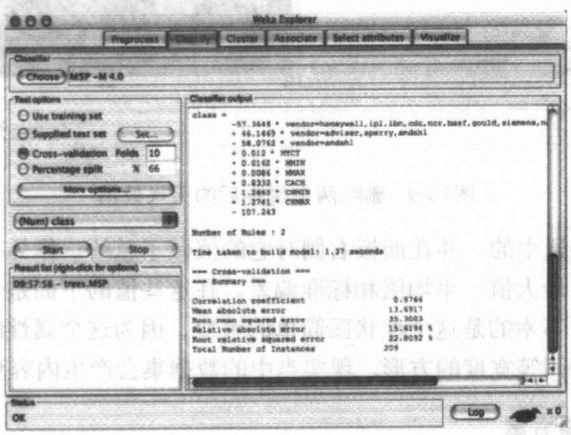
举例来说，图10-10a的表1-5中的CPU性能数据集已载入Weka。右下角显示的是第一个属性*vendor*值的柱状图。在图10-10b中，进入分类面板，单击左上方的*Choose*按钮，打开如图10-4a所示的等级式菜单中的树的部分，找到*M5P*，然后单击*Start*，即可将模型树归纳器M5'
选定为分类器。通过把具有共同功能性的项目组合起来形成的等级式结构有助于快速发现具体的分类器。 382~384

图10-11给出了输出结果。修剪过的模型树只是一个简单的，根据属性MMAX进行分割的决策树桩（decision stump），再加上分别对应着一个叶的两个线性模型。两个模型都涉及到一个名词性属性vendor，以及一些数值性属性。表达式vendor=adviser, sperry, amdahl可做如下

解读：如果vendor是adviser，sperry或amdahl中的一个，就替换1，否则就替换0。模型树描述的后面是几个衡量该模型性能的指标。由图10-10b中的10折交叉验证（非分层的，因为对于数值预测毫无意义）测试选项得出的。第5.8节（表5-8）解释了不同测量方式的含义。



a)



b)

385 图10-10 用M5'处理CPU性能数据

普通线性回归（第4.6节），是另外一种用于数值预测的方案，可在图10-4a菜单中的功能部分的线性回归类别下找到。它只建立一个单独的线性回归模型而不是图10-11中见到的两个；当然，这个模型的性能也要差一些。

要想对它们各自的性能做个比较，我们把它们所建立的方案的误差可视化，就像我们在图10-6b中对鸢尾花数据集所进行的操作一样。右键单击历史列表中的项目，然后选择*Visualize classifier errors*（可视化分类器误差）得到如图10-12所示的二维数据点阵。图中的点呈现不同的颜色，分别代表不同的类，但对图中的例子来说，因为类是数值性的，所以颜色变化是连续的。在图10-12中，X坐标轴选择了属性vendor，Y坐标轴是所选实例的数量，这样选择坐标会使这些点的分布较好。每个数据点都标示为一个交叉符，交叉符的大小代表了该实例的误差的绝对值。与图10-12b（线性回归）中的交叉符相比较，图10-12a（M5'）中的交叉符尺寸较小，这表明M5'更有优势。

```
=== Run information ===

Scheme:       weka.classifiers.trees.M5P -M 4.0
Relation:     cpu
Instances:    209
Attributes:   8
              vendor
              MYCT
              MMIN
              MMAX
              CACH
              CHMIN
              CHMAX
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

M5 pruned model tree:
(using smoothed linear models)

MMAX <= 14000 : LM1 (141/4.178%)
MMAX >  14000 : LM2 (68/50.073%)

LM num: 1
class =
    -2.0542 *
    vendor=honeywell,ipl,ibm,cdc,ncr,basf,gould,siemens,nas,adviser,sperry,amdahl
    + 5.4303 * vendor=adviser,sperry,amdahl
    - 5.7791 * vendor=amdahl
    + 0.0064 * MYCT
    + 0.0016 * MMIN
    + 0.0034 * MMAX
    + 0.5524 * CACH
    + 1.1411 * CHMIN
    + 0.0945 * CHMAX
    + 4.1463

LM num: 2
class =
    -57.3649 *
    vendor=honeywell,ipl,ibm,cdc,ncr,basf,gould,siemens,nas,adviser,sperry,amdahl
    + 46.1469 * vendor=adviser,sperry,amdahl
    - 58.0762 * vendor=amdahl
    + 0.012 * MYCT
    + 0.0162 * MMIN
    + 0.0086 * MMAX
    + 0.8332 * CACH
    - 1.2665 * CHMIN
    + 1.2741 * CHMAX
    - 107.243

Number of Rules : 2
```

图10-11 用于数值性预测的M5'程序的输出结果

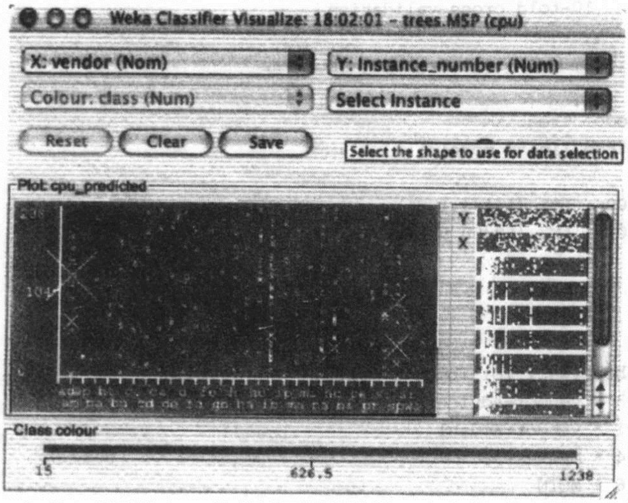
```
Time taken to build model: 1.37 seconds

=== Cross-validation ===
=== Summary ===

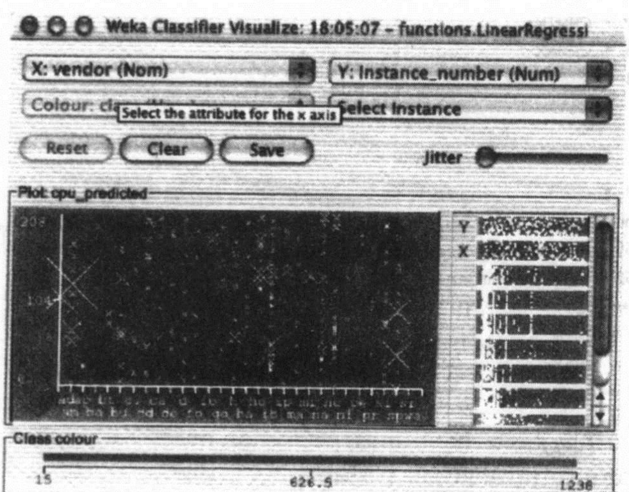
Correlation coefficient                  0.9766
Mean absolute error                     13.6917
Root mean squared error                 35.3003
Relative absolute error                 15.6194 %
Root relative squared error             22.8092 %
Total Number of Instances              209
```

386~387

图 10-11（续）



a) 由M5'得到



b) 由线性回归得到

图10-12 可视化误差

10.2.3 自己动手：用户分类器

用户分类器（在第3.2节的结尾部分提到过）使用户得以互动式地建立它们自己的分类器。用户分类器放在图10-4a中的等级式菜单里的树的部分，在*UserClassifier*的下面。我们用一个新问题来演示它的操作，根据平均密度，色调，尺寸，位置和不同的简单质地特征等属性，将视觉图像数据划分成类似于grass，sky，foliage，brick和cement（草，天空，树叶，砖和水泥）等类。Weka发行配套包提供一个称作*segment-challenge.arff*的训练数据文件。该文件载入后，再选择用户分类器。在*Classify*面板上，选择一个名字叫*segment-test.arff*的特别训练测试集作为*Supplied test set*（提供的测试集）用来评估。当用户必须手动为每个折构建分类器时，交叉验证评估是做不到的。 388

单击*Start*后，一个新窗口出现，此时Weka等待用户建立分类器。*Tree Visualizer*（树可视化器）和*Data Visualizer*（数据可视化器）标签用于在不同的查看方式间切换。前者显示的是分类树的当前状态，并且每个节点给出了该节点上属于每一类的实例数。其目的是为了得到一个叶节点越纯越好的树。一开始当然只有一个节点，根，所有数据都包含在其中。然后切换到*Data Visualizer*，生成一个分割。这样就得到了我们在图10-6b中通过鸢尾花数据集和图10-12中通过CPU性能数据看到的同样的二维点阵。像前面一样，分别为X和Y坐标选择属性。在这里，用来挑选的标准是找出一个能使类与类分离的尽可能干净的X和Y的组合。图10-13a给出了一个好选择：区域－质心－行作为X及强度－平均值作为Y。

找出一个好的分离后，用户必须在图表中指定一个区域。在Y轴选择器下方的下拉菜单中可找到四个工具。*Select Instance*（选择实例）确认一个具体的实例。*Rectangle*（矩形，见图10-13a）使用户得以在图表上拖出一个矩形。利用*Polygon*和*Polyline*（多边形和多边线），用户可构建任意形状的多边形或画任意形状的多边线（左键单击添加一个角的顶点，右键单击完成操作）。一个区域一旦被选定，就变成灰色。图10-13a中，用户定义了一个矩形。*Submit*按钮会在树中创建两个新节点，一个节点含有那些选定的实例，另外一个包含全部其余实例。*Clear*会清除所做的选择；*Save*将当前树节点上的实例以ARFF文件的方式存储起来。

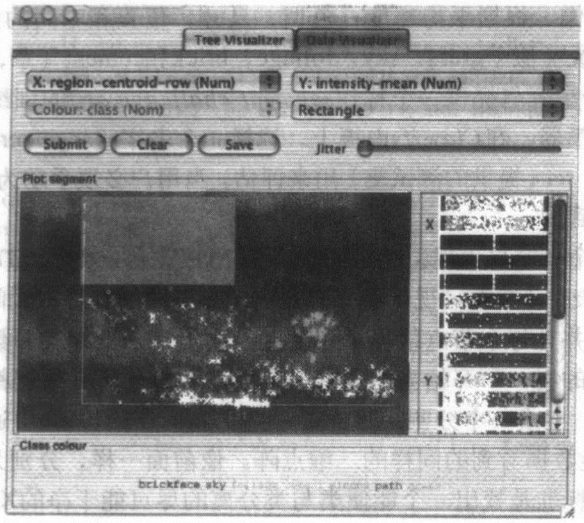
此时，树可视化器在图10-13b中显示了所构建的树。其中一个节点（左节点）对类sky来说是纯的（pure），但另一个（右节点）则是混合节点，需进一步分割。单击哪个节点，则整个数据集中与该节点相关联的子集（subset）就会被数据可视化器可视化。继续增加节点，直到用户对所得结果满意，即，几乎所有节点都是纯的为止。然后在树可视化器的空白处右键单击，选择*Accept the Tree*（接受此树）。Weka用测试集评估用户的树并输出性能统计数据（对于所举例子，80%的结果是好的）。

使用手动方式构建树非常繁琐。但Weka可以在任何节点下面构建子树，从而替用户完成作业，只需右键单击该节点即可。

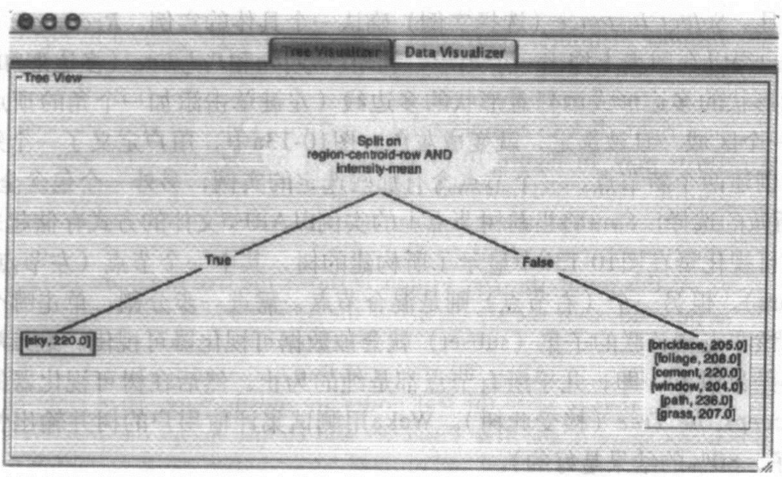
10.2.4 使用元学习器

元学习器（第7.5节）可将简单的分类器变成功能更强大的学习器。例如，要想提升探索者的决策树桩的性能，打开*Classify*面板，在等级式菜单中的meta部分选择*AdaboostM1*分类器。当用户双击该分类器对其进行设置时，图10-14所示的对象编辑器会出现。这个分类器有自己的分类器域，我们设为*DecisionStump*（见图）。这个域设定法本身也可以通过双击的方式进行设置（本例是个例外，*DecisionStump*碰巧没有可编辑的属性）。单击OK回到*Classify*主面板， 389

单击*Start*可尝试着将决策树桩的性能提升至最高10倍。最终的结果是鸢尾花数据的150个实例中只有7个被错误标志。考虑到这仅仅是一个初级决策树桩以及如此少的提升循环次数，它的性能的确不错。



a) 数据可视化器



b) 树可视化器

图10-13 在划分过的数据上运行用户分类器

10.2.5 聚类和关联规则

通过*Cluster*和*Associate*面板实现聚类算法（第6.6节）及启动用于找出关联规则的方法（第4.5节）。在对数据集进行聚类时，Weka会显示聚类的数量和每个聚类所含的实例数。对某些算法来说，聚类的数量可在对象编辑器中通过设定参数的方式指定。如果使用的是概率性聚类方式，Weka测量的是对应训练数据的聚类的对数似然：这个对数似然越大，说明所建立的模型与数据拟合得越好。增加聚类的数量一般来说会增加对数似然，但会导致过度拟合。

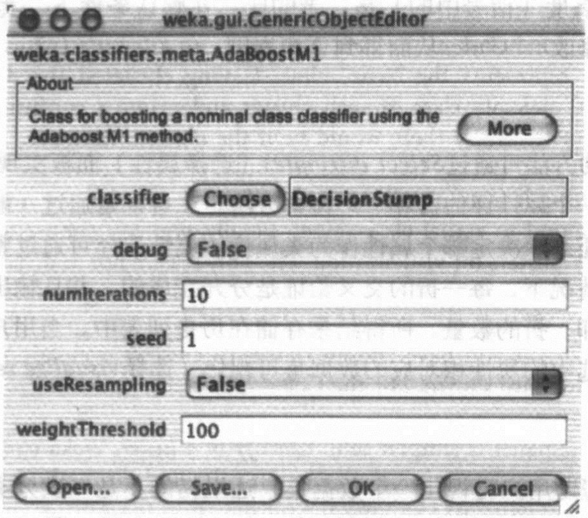


图10-14 设置一个用于提升决策树树桩性能的元学习器

*Cluster*面板上的操纵元件与*Classify*面板上的相类似。用户可指定一些同样的评估方法，使用训练集，提供测试集，百分比分割（后两项用于对数似然）。更进一层的手段，类对应聚类的评估，它比较的是所选定的聚类与数据中预先指定的类匹配程度的好坏。用户选择一个属性作为“正确”的类（它必须是名词性的）。对数据进行聚类后，Weka会确定每个聚类中的多数类并给出一个混淆矩阵，表明如果使用聚类而不是用正确的类会有多少误差。如果数据集中有一个属性是类属性，用户可在聚类的过程中通过在一个含有这些属性的下拉菜单中将其选定的方式把它忽略掉，然后观察这些聚类与实际类值的对应程度。最后，用户可选择是否将这些聚类存储起来用于可视化。选择不要存储的唯一理由是为了节省空间。对于分类器来说，右键单击结果列表使其可视化的好处是用户可查看类似于图10-6b中的二维散点图。如果用户决定使用类对应聚类的评估，类的指定误差也会列出来。对于Cobweb聚类方案，用户也可以将树可视化。 390~391

*Associate*面板要比*Classify*或*Cluster*面板简单。Weka中有三种算法可用来决定关联规则，但没有手段评估这些规则。图10-15给出了利用Apriori程序从天气数据的名词性版本中寻找关联规则（第4.5节所描述的）的输出结果。尽管数据比较简单，还是找到了一些规则。箭头前面的数字表示的是箭头前面的前提条件为真的实例数，箭头后面的数字代表箭头后面的结论

```
 1. outlook=overcast 4 ==> play=yes 4    conf:(1)
 2. temperature=cool 4 ==> humidity=normal 4    conf:(1)
 3. humidity=normal windy=FALSE 4 ==> play=yes 4    conf:(1)
 4. outlook=sunny play=no 3 ==> humidity=high 3    conf:(1)
 5. outlook=sunny humidity=high 3 ==> play=no 3    conf:(1)
 6. outlook=rainy play=yes 3 ==> windy=FALSE 3    conf:(1)
 7. outlook=rainy windy=FALSE 3 ==> play=yes 3    conf:(1)
 8. temperature=cool play=yes 3 ==> humidity=normal 3    conf:(1)
 9. outlook=sunny temperature=hot 2 ==> humidity=high 2    conf:(1)
10. temperature=hot play=no 2 ==> outlook=sunny 2    conf:(1)
```

图10-15 关联规则Apriori程序的输出结果

也为真的实例数；置信度（括号中的）是二者的比。在默认条件下，共发现了10条规则：用户可通过对象编辑器修改*numRules*从而得到更多规则。

10.2.6 属性选择

有几种属性选择的方法可通过*Select attributes*（选择属性）面板实现。正如在第7.1节中解
392 释过的，这会涉及到一个属性评估器和一个搜索方法。二者都是通过一般途径选定，且在对象编辑器中设置。用户还必须决定哪个属性作为类属性。属性选择可通过整个训练集或通过交叉验证做出。在后一种情况下，每一折的交叉验证是分开完成的，并且输出结果会显示出每个属性被选中了多少次，即，折的数量。所得结果存储在历史列表中。当用户右键单击历史列表中的条目时，可将与被选中的属性相对应的数据集可视化（选择*Visualize reduced data*）。

10.2.7 可视化

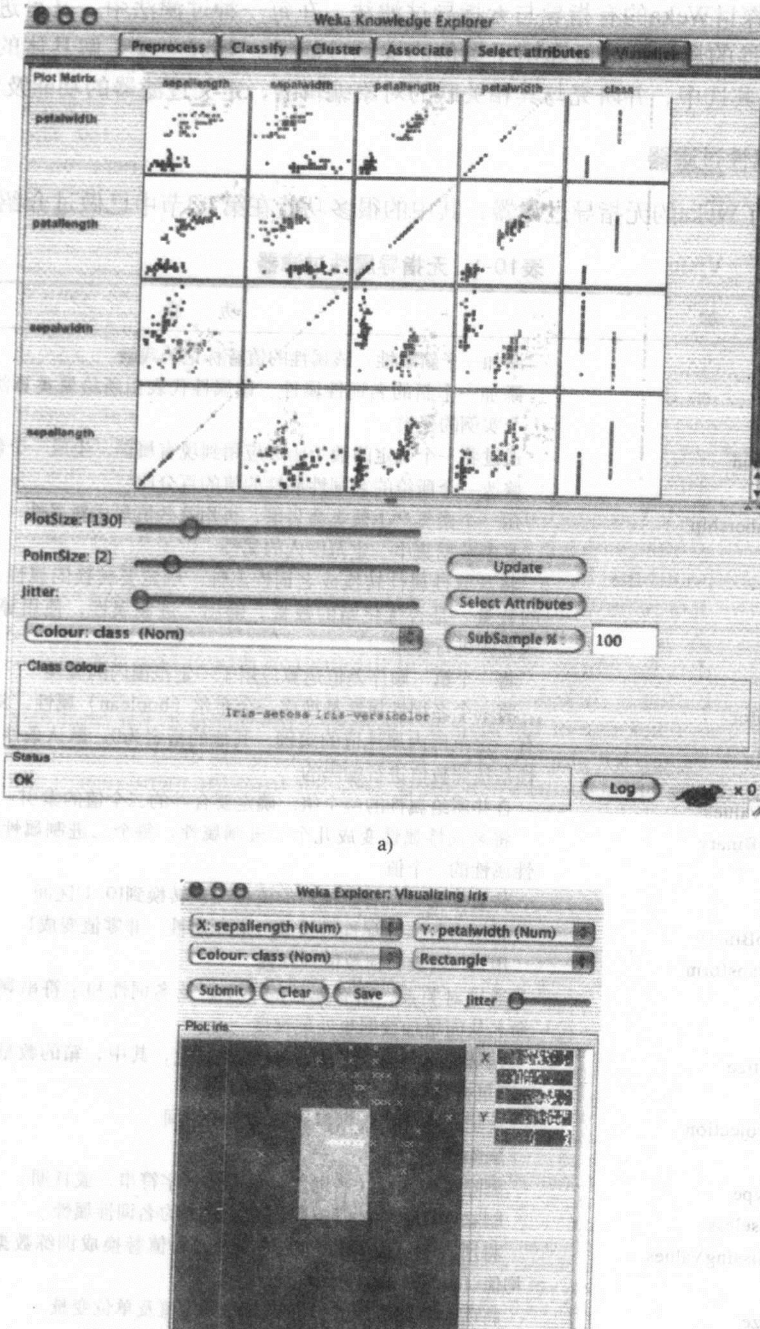
*Visualize*面板可帮助用户可视化一个数据集，这里指的不是一次分类或一个聚类模型的结果，而是数据集本身。它显示的是每对属性的一个二维散点图。图10-16a给出了鸢尾花数据集的可视化结果。用户可选择一个属性，通常是所选定的类属性，用面板底部的控制元件为代表数据的点加上颜色。如果属性是名词性的，颜色是离散的；如果是数值性的，则色谱从蓝色（较小值）到橙色（较大值）呈连续分布。没有类值的数据的点用黑色表示。用户可改变每个点阵的大小和点阵中点的大小，以及抖动度。抖动是应用于X和Y坐标值的随机位移，目的是使互相重叠的点分开。如果没有抖动，位于同一个数据点上的上千个实例看起来就像一个。用户可选择一定的属性来减小这个点图矩阵的尺寸，并对数据进行子采样以提高效率。在面板上所做的改动只有点击*Update*按钮后才能生效。

单击矩阵中的任何一个散点图使其放大。例如，单击左上角的点图得到如图10-16b所示面板。用户可放大该面板中的任何区域，方法是在右上方的菜单中选择*Rectangle*，并按照图中显示的那样拉出一个矩形。左上方的*Submit*（提交）按钮会按照比例在查看区域中重新调整这个矩形框。

10.3 过滤算法

现在我们来深入探讨Weka中实现的各种过滤算法。在探索者中可运用这些算法，在第11、12章将描述的知识流和实验者界面中也可以。所有的过滤器都对输入的数据集进行某种转换。当一个过滤器通过*Choose*按钮选中时，它的名字会出现在按钮旁边的横栏中。单击这个横栏，弹出一个通用对象编辑器确定该横栏的属性。横栏中出现的是该过滤器的命令行版本，减号后面的字母指定该过滤器的参数。这是一个学习如何直接使用Weka命令的好方法。

过滤器有两种：有指导（supervised）和无指导（unsupervised）（第7.2节）。这个似乎不
393 经意的区别实际上却掩盖了一个根本课题。过滤器通常应用于训练数据集，再应用于测试文件。如果过滤器是有指导的，比方说，假如它使用类值以取得好的离散区间，那么将该过滤器应用于测试数据则会对所得结果造成偏差。应用于测试数据的必须是由训练数据取得的离散区间。当使用有指导过滤器时，用户必须小心确保公正地评估所得结果。使用无指导过滤器则不存在这样的问题。



a)

b)

图10-16 可视化鸢尾花数据集

我们会分别探讨Weka的有指导与无指导过滤法。在每一种过滤法中，又可进一步区分为用于数据集中属性的属性过滤器和用于实例的实例过滤器。要进一步了解具体的过滤器，在Weka探索者中将其选中，并研究与其相关联的对象编辑器，定义过滤器的功能及各种参数。

10.3.1 无指导属性过滤器

表10-1列出了Weka的无指导过滤器。其中的很多功能在第7.3节中已做过介绍。

表10-1　无指导属性过滤器

名　称	功　能
Add	添加一个新属性，该属性的值皆标记为残缺
AddCluster	添加一个新的名词性属性，该属性代表由所给聚类算法指定到每一个实例的聚类
AddExpression	通过将一个确定的数学函数应用到现有属性，生成一个新的属性
AddNoise	修改一个所给的名词性属性的值的百分比
ClusterMembership	用一个聚类产生聚类会员值，再用这些值形成新属性
Copy	复制数据集中一定范围内的属性
Discretize	将数值性属性转换成名词性属性：确定要转换的属性、箱（bin）的数量，是否优化箱的数量，输出二进制属性，使用等宽（默认）或等频进行装箱
FirstOrder	将一个第一顺序差值运算应用于一定范围内的属性
MakeIndicator	将一个名词性属性替换成一个布尔（boolean）属性。将1指定给具有一定范围内属性值的实例，其他的指定为0。默认条件下，布尔属性是按照数值进行编码的
MergeTwoValues	合并所给属性的两个值：确定要合并的二个值的索引
NominalToBinary	将名词性属性变成几个二进制属性，每个二进制属性对应该名词性属性的一个值
Normalize	将数据集中所有数值性的值按比例转换到[0, 1]区间
NumericToBinary	将所有数值性属性转换成二进制属性：非零值变成1
NumericTransform	用Java函数转换数值性属性
Obfuscate	通过对关系，所有属性的名字，及名词性和字符串属性值进行重命名从而增加数据集处理难度
PKIDiscretize	用等频区间装箱法离散数值性属性，其中，箱的数量等于属性值（不包括残缺值）的数量的平方根
RandomProjection	用随机矩阵将数据投影到低维子空间
Remove	删除属性
RemoveType	删除某种属性（名词性，数值性，字符串，或日期）
RemoveUseless	删除常量属性，以及变化幅度过大的名词性属性
ReplaceMissingValues	将所有名词性和数值性属性的残缺值替换成训练数据的众数和平均值（modes和means）
Standardize	将所有数值性属性标准化为零平均值及单位变量
StringToNominal	将字符串属性转换为名词性属性
StringToWordVector	将字符串属性转换为代表词的出现频率的向量：用户可选择分隔符，且可选范围很广
SwapValues	将一个属性的两个值对调
TimeSeriesDelta	将当前实例的属性值替换成当前值与以前（或将来）的实例的值的差
TimeSeriesTranslate	将当前实例的属性值替换成与以前（或将来）的实例的等同值

添加和删除属性

*Add*在指定位置插入一个属性，该属性的值已声明对所有实例皆为残缺。使用通用对象编辑器确定属性的名字，在编辑器中会出现一串属性及属性的可能的值（对名词性属性来说）。*Copy*复制现有的属性，以便当用户试验可覆盖属性值的过滤器时可以将这些属性值保存下来。利用一个表达式则可将几个属性一起进行复制，例如，1-3指前三个属性，或者first-3,5,9-last指属性1，2，3，5，9，10，11，12，…。这些选项也可表示相反的含义，即只对那些未在表达式中指明的属性起作用。很多过滤器都具备这些特色。

*Remove*前面已讨论过。类似的过滤器有*RemoveType*，删除一个指定种类（名词性，数值性，字符串，或日期）的所有属性，和*RemoveUseless*，删除常量属性和那些对几乎所有实例来说其值都不相同的名词性属性。用户可通过确定不相同的值的数量作为值的总数的百分比来决定允许偏差，决定是否删除一个属性。如果*Preprocess*面板上的菜单用来设定一个类属性，一些无指导属性过滤器运行起来会不同。例如，*RemoveType*和*RemoveUseless*两者都会跳过类属性。

*AddCluster*将一种聚类算法应用于数据，然后再将数据过滤。用户通过对象编辑器选择聚类算法。聚类器的设置方式与过滤器一样（第10.6节）。*AddCluster*对象编辑器通过自身的*Choose*按钮用来选择聚类器，用户可通过单击按钮旁的横栏得到另外一个对象编辑器，并在其面板上对该聚类器进行配置，该面板必须填写完整后才能回到*AddCluster*对象编辑器。比起在书中阅读这些说明，也许用户在实践中亲手操作一遍会更容易理解。不论怎样，一旦用户选定一个聚类器，*AddCluster*会用它来为每个实例指定一个聚类的号码，作为该实例的一个新属性。当用户对数据进行聚类时，对象编辑器还允许用户忽略一些属性，如前面谈到*Copy*时所描述过的。我们在描述过滤器对象编辑器时也探讨过，*ClusterMembership*利用一个聚类器产生会员值。对每个实例来说，都有一个以这些会员值为属性的新版本实例生成。如果设定了类属性，则对类属性不作修改。

*AddExpression*通过将一个数学函数应用于数值性属性而生成一个新的属性。表达式可含有属性索引和常量，四则运算符+，－，*，/，和^；函数*log*和*exp*，*abs*和*sqrt*，*floor*，*ceil*和*rint*[⊖]，*sin*，*cos*和*tan*；以及括号。属性是通过加前缀*a*确定的，例如*a*7指第7个属性。以下是一个表达式范例

$$a1\verb|^|2*a5/\log(a7*4.0)$$

有一个纠错选项可用来将新属性的值替换成所给的表达式的后缀解析式。

与*AddExpression*应用数学函数不同，*NumericTransform*所做的是通过应用一个给定的Java函数对选定的数值性属性进行任意转换。任何函数都可以，只要它以一个双精度浮点数为参数并返回另外一个双精度浮点数，例如，*java.lang.Math*中的*sqrt*()。其中一个参数是Java中类的名字，该类中实现了这个函数（必须是完全符合规范的名字）；另外一个参数是转换方法本身的名字。 394~397

*Normalize*将数据集中所有数值性的值按比例转换到0～1之间。*Standardize*则对它们进行转换，取得零平均和单位变量。二者都跳过类属性，如果有的话。

⊖ *rint*函数将一个数进位成最接近的整数。

修改值

*SwapValues*将一个名词性属性的两个值进行对换。两个值的先后顺序完全是表面的，对学习毫无影响，但是，如果类属性被选中，则改变顺序会影响混淆矩阵的配置。*MergeTwoValues*将一个名词性属性的两个值合并为一个单独的类别。新值的名字由原来的两个值串接而成，且所有出现原来的两个值的地方都替换成新值。新值的索引是原来的两个值的索引中较小的一个。例如，如果用户合并天气数据中属性阴晴的前两个值，其中共有5个sunny，4个overcast和5个rainy实例，新的阴晴属性的值则变成sunny-overcast和rainy；且将有9个sunny-overcast实例再加上原来的5个rainy实例。

处理残缺值的一个办法是在应用学习方案前将它们全面替换。*ReplaceMissingValues*将每个残缺值替换成数值性属性值的平均值和名词性属性值的众数。如果一个属性被设为类，其残缺值不做任何替换。

转换

很多过滤器把属性从一种形式转换成另外一种。*Discretize*使用等宽或等频区间装箱法（第7.2节）离散一定区间内的，通过正常方式指定的数值性属性。对前一种方法来说，装箱的数量可通过指定，或通过使用留一交叉验证使可能性最大化的方式来自动选定。*PKIDiscretize*用等频区间装箱法离散数值性属性，其中装箱的数量是值（不包括残缺值）的数量的平方根。这两个过滤器都跳过类属性。

*MakeIndicator*可将一个名词性属性变成一个二进制的指示器属性，且可用于把一个多类的数据集变成几个二类的数据集。它把选定的名词性属性替换成一个二进制属性，对每个实例，如果该属性对应的原始值存在的话，所选名词性属性的值设为1，否则为0。新属性在默认条件下声明为数值性属性，但也可以声明为名词性的。

一些学习方案，如支持向量机，仅处理二进制属性。*NominalToBinary*过滤器将一个数据
398 集中所有的具有多个值的名词性属性转换成二进制属性。即，以一对一的简单编码方式，将每个有k个值的属性以k个二进制属性替换。原本就是二进制的属性不做任何改变。*NumericToBinary*把所有数值性属性转换成名词性二进制属性（如果是类属性，则不做转换）。如果数值性属性的值是零，新属性也是零；如果是残缺值，新属性也标记为残缺；否则，新属性的值是1。这些过滤器同样跳过类属性。

*FirstOrder*将N个一定区间内的数值性属性替换成$N-1$个数值性属性，新属性的值将是原来实例的连续顺位属性值的差。举例说明，如果原来的属性值是3，2和1，新值将是-1和-1。

字符串转换

字符串属性的值的数量是不确定的。*StringToNominal*将字符串属性转换成具有一定数量的值的名词性属性。用户应确保所有将要出现于潜在的测试数据中的字符串值在数据集中都有所体现。

*StringToWordVector*生成代表字符串中每个词出现的频率的属性。这组词即这些新生成的属性集，是由原来的数据集决定的。默认条件下，每个词都变成一个属性，其值为1或0，分别表示该词在字符串中出现与否。新属性可使用用户决定的前缀来命名，以使这些由不同的字符串属性派生来的属性相互区分开来。

有很多选项可影响到tokenization。字可由连续的字母序列形成，或按照一组给定的分隔符进行分离。在加入字典前会转换成小写字母，或干脆忽略掉预先确定的所有英语停止词

（stopwords）列表中的词。在前k个按照出现频率排序的词中所没有的词会被删除（如果排序时在第k个位置出现并列，保留词的数量比k略多一点儿也是可以的）。如果指定了一个类属性，每个类的前k个词要保留。每个词属性的值反映的是该词在字符串中是否出现，但这是可以修改的。一个词在字符串中出现的次数也可作为属性值使用。词的频率可被正常化以使每个文本的属性向量取得同样的欧几里得长度，该长度不可是1，而应是被用作原始字符串属性的所有文本的平均长度，以避免出现非常小的数字。另外一种方法是，词i出现在文本j中的频率f_{ij}可通过log（$1+f_{ij}$）或TF × IDF测量（第7.3节）进行转换。

时间系列

有两个过滤器可处理时间系列数据。*TimeSeriesTranslate*将当前实例的一个属性（或一些属性）的值替换成其他（以前的或将来的）实例的等同的值。*TimeSeriesDelta*将当前实例的属性值替换成当前值和其他实例的值的差。在以上两种情况下，时间位移（time-shifted）值未知的实例可被删除、或用残缺值代替。 399

随机化

其他属性过滤器用来降低（degrade）数据的质量。*AddNoise*修改一个名词性属性的值的百分比。残缺值可被保留，或伴随其他值一起修改。*Obfuscate*通过对关系，属性的名字，和名词性及字符串属性的值进行重新命名来隐匿数据。*RandomProjection*用随机的具有单位长度的列的矩阵将数据集投影到低维子空间（第7.3节）。类属性不包含在投影中。

10.3.2 无指导实例过滤器

Weka的实例过滤器见表10-2，会影响到数据集中所有的实例，而不是一个或一些属性的所有值。

随机化和子采样

用户可随机化数据集中的实例顺序。*Normalize*把所有数值性的属性（不包括类属性）看作一个向量并将其正常化为一个给定的长度。用户可指定所用的向量的长度和范数。

表10-2　无指导实例过滤器

名　称	功　能
NonSparseToSparse	将所有输入的实例转换为稀疏模式（第2.4节）
Normalize	把数值性属性看成一个向量并将其正常化为一个给定的长度
Randomize	随机化数据集中实例的顺序
RemoveFolds	输出数据集的一个指定的交叉验证的折
RemoveMisclassified	删除那些按照指定的分类器未能正确分类的实例，有效删除例外
RemovePercentage	对一个数据集按照一个给定的百分比进行删除
RemoveRange	从一个数据集中将一定区间内的实例删除
RemoveWithValues	滤除具有特定属性值的实例
Resample	用替代式采样法，由一个数据集生成一个随机的子采样
SparseToNonSparse	将所有输入的稀疏实例转换为非稀疏格式

有许多不同的方法可产生数据的子集。用*Resample*通过替代式采样法产生一个随机样本，或用*RemoveFolds*将数据分割为给定数量的交叉验证的折并降低到仅仅一个折。如果给定一个随机数的种子，则可对数据集进行随机组合，然后再抽取子集。*RemovePercentage*删除一定

百分比的实例，而*RemoveRange*删除一个特定区间内的实例数。要想删除所有含有某些特定的名词性属性值的实例，或者对于数值性属性来说，其值大于或小于某个阈值的实例，用*RemoveWithValues*。默认条件下，呈现为一组指定名词性属性的值（如果指定的属性是名词性的）中的一个值的所有实例，或其属性值低于一个阈值（如果是数值性属性）的所有实例都会被删除。然而，以上吻合标准也可以反向适用。

用户可通过将一个分类方法应用于一个数据集来删除数据集中超出合理范畴的值（确定分类方法的方式与上面谈到*AddCluster*时确定聚类方法一样），并使用*RemoveMisclassified*删除被错误分类的实例。

稀疏实例

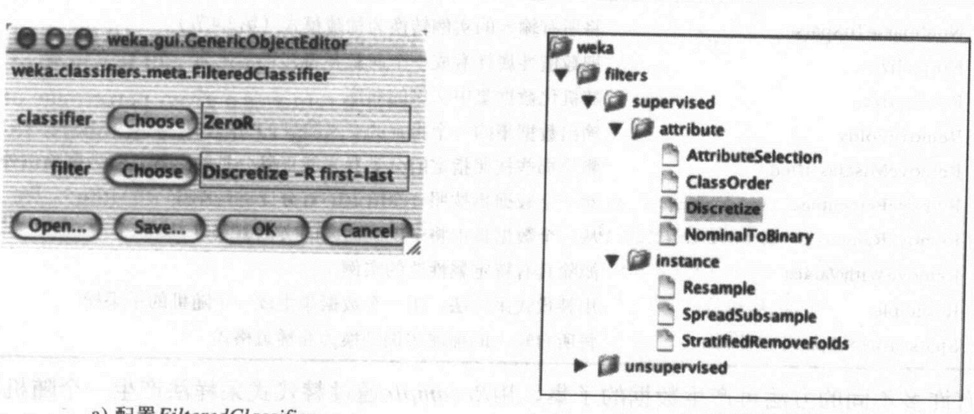
*NonSparseToSparse*和*SparseToNonSparse*过滤器可用来在数据集的正常形式与它的稀疏形式（见第2.4节）之间进行转换。

10.3.3　有指导过滤器

与无指导过滤器一样，有指导过滤器也可在探索者的*Preprocess*面板上找到。用户需小心，尽管它们看上去像是预处理操作，但实际上不是。我们在上面谈到离散的时候提到了到这一点，测试数据的分割一定不能用测试数据的类值，因为这些类值是假定未知的，而且一般来说对有指导过滤器也是如此。

因为受欢迎的原因，Weka允许用户像调用无指导过滤器一样调用有指导过滤器作为一种预处理操作。然而，如果用户想把它们用作分类，则需采用一种不同的方式。Weka提供了一个元学习器用于调用一个过滤器，从而将学习算法包含在过滤机制中。这样就可以用由训练数据生成的过滤器来过滤测试数据。这对一些无指导过滤器也很有用。例如，在*StringToWordVector*中，字典仅仅是由训练数据生成的，测试数据中的异常字将被删除。欲以该种方式使用有指导过滤器，调用菜单的meta部分中的*Filtered Classifier*元学习方案，该菜单
400~401 可通过单击*Classify*面板上的*Choose*按钮显示。图10-17a给出了该元学习方案的对象编辑器。用户可在对象编辑器中选择分类器和过滤器。图10-17b列出了过滤器菜单。

与无指导过滤器一样，有指导过滤器可分为属性过滤器和实例过滤器，见表10-3和表10-4。



a) 配置*FilteredClassifier*　　　　b) 过滤器菜单

图10-17　用Weka的元学习器做离散

表10-3 有指导属性过滤器

名　称	功　能
AttributerSelection	像Select attributes面板一样，提供采用相同的属性选择方法
ClassOrder	随机化或修改类值的顺序
Discretize	将数值性属性转换为名词性属性
NominalToBinary	将名词性属性转换为二进制属性，如果类是数值性的，用有指导方法

表10-4 有指导实例过滤器

名　称	功　能
Resample	用替代式采样法，生成一个数据集的随机子采样
SpreadSubsample	用替代式采样法，生成一个随机的子采样，该子采样在类频率之间具备一个给定的分布
StratifiedRemoveFolds	输出数据集的一个确定的分层交叉验证折

有指导属性过滤器

Discretize（图10-17高亮部分）使用的是有指导离散的MDL方法（第7.2节）。用户可指定一定区间内的属性，或将离散的属性硬性地当作二进制属性。类属性必须是名词性的。在默认情况下通常使用Fayyad和Irani标准（1993年），但Kononenko方法（1995年）是一个选项。

*NominalToBinary*过滤器有一个有指导版本，它把所有多值的名词性属性转换为二进制属性。在这个版本中，该转换取决于其中的类是数值性的还是名词性的。如果是名词性的，前面用过的方法同样适用，一个具有k个值的属性转换为k个二进制属性。但如果类是数值性的，采用的则是第6.5节中描述过的方法。不论哪种情况，类属性本身不做转换。 402

*ClassOrder*修改类值的排序方法。用户决定新的排序是随机的，还是按类频率由高至低或由低至高排列。该过滤器绝对不可以与*Filtered Classifier*元学习方案一起使用！*AttributeSelection*可用于自动属性选择，并可提供与探索者的*Select attribute*面板（后面会谈到）同样的功能。

有指导实例过滤器

共有三种有指导实例过滤器。*Resample*，顾名思义，除了可完成子采样中的类分布外，它就是一个无指导过滤器。除此以外，它还可以经过适当设置，从而在类分布中加入偏差使其成为平均分布。*SpreadSubsample*还生成一个随机的子采样，但最少见与最常见之间的频率差是可以控制的，比方说，用户可在类频率中指定一个2:1的差。与无指导实例过滤器*RemoveFolds*一样，*StratifiedRemoveFolds*输出数据集中一个指定的交叉验证的折。所不同的是，此次的折是分层的。

10.4 学习算法

在*Classify*面板上，当用户通过*Choose*按钮选择一个学习算法时，相关分类器的命令行版本，包括用减号标注的参数，会出现在按钮旁的横栏中。要对这些参数进行修改，单击该横栏弹出一个相应的对象编辑器。表10-5列出了Weka中的分类器。这些分类器被划分为贝叶斯（Bayesian）分类器，树，规则，函数，懒惰分类器以及一个最终的杂项类分类器。我们在对这些分类器及其参数做简单描述。欲对它们做深入了解，可在Weka的探索者界面上任选一种

分类器，然后根据所得的相应对象编辑器进行研究。更进一层的分类器元学习器，会在下一节中描述。

表10-5　Weka中的分类器算法

名称		功能
Bayes	AODE	平均单一依赖估计器
	BayesNet	学习贝叶斯网
	ComplementNaiveBayes	建立一个补偿的朴素贝叶斯分类器
	NaiveBayes	标准概率朴素贝叶斯分类器
	NaiveBayesMultinomial	朴素贝叶斯的多重名词性版本
	NaiveBayesSimple	朴素贝叶斯的简单实现
	NaiveBayesUpdateable	每次只学习一个实例的递增朴素贝叶斯分类器
Trees	ADTree	建立交互决策树
	DecisionStump	建立单级决策树
	Id3	基本的分治决策树算法
	J48	C4.5决策树学习器（实现C4.5revision8）
	LMT	建立logistic模型树
	M5P	M5'模型树学习器
	NBTree	在树的叶端用朴素贝叶斯分类器建立一个决策树
	RandomForest	构建随机森林
	RandomTree	构建一个每个节点含有指定数量的随机属性的树
	REPTree	使用减少－误差修剪的快速树学习器
	UserClassifier	允许用户建立自己的决策树
Rules	ConjunctiveRule	简单结合规则学习器
	DecisionTable	建立一个简单的决策表多数分类器
	JRip	用于快速，有效的规则归纳的RIPPER算法
	M5Rules	从利用M5'建立的模型树中获取规则
	Nnge	使用非嵌套泛化的样本集来产生规则的最近邻方法
	OneR	1R分类器
	Part	从利用J4.8建立的部分决策树中获取规则
	Prism	应用于规则的简单覆盖算法
	Ridor	Ripple-down规则学习器
	ZeroR	预测多数类（如果是名词性的）或平均值（如果是数值性的）
Functions	LeastMedSq	使用中间值而非平均值的稳健回归
	LinearRegression	标准线性回归
	Logistic	建立线性logistic回归模型
	MultilayerPerceptron	反向传播的神经网络
	PaceRegression	用Pace回归建立线性回归模型
	RBFNetwork	实现一个径向基函数网络
	SimpleLinearRegression	学习一个基于单个属性的线性回归模型
	SimpleLogistic	使用已有的属性选择，建立线性logistic回归模型
	SMO	用于支持向量分类的连续最小优化算法
	SMOreg	用于支持向量回归的连续最小优化算法
	VotedPerceptron	投票感知器算法
	Winnow	成倍更新的，由错误驱动的感知器

（续）

名称		功能
Lazy	IB1	基本的，基于实例的最近邻学习器
	IBk	k最近邻分类器
	KStar	使用泛化距离函数的最近邻
	LBR	懒惰贝叶斯规则分类器
	LWL	用于局部加权学习的一般算法
Misc.	Hyperpipes	在实例空间中基于超大量实例的极简单的快速学习器
	VFI	投票特征区间方法、简单快速

10.4.1 贝叶斯分类器

*NaiveBayes*实现概率的朴素贝叶斯分类器（第4.2节）。*NaiveBayesSimple*用正态分布来构建数值性属性的模型。*NaiveBayes*可利用核密度估计器，从而在正常化假定总体上不正确的情况下改进性能。它还可用有指导离散来处理数值性属性。*NaiveBayesUpdateable*是一个每次只处理一个实例的递增版本；利用核估计器，但无法使用离散。*NaiveBayesMultinomial*实现多重名词性贝叶斯分类器（第4.2节）。*ComplementNaiveBayes*建立一个增补的朴素贝叶斯分类器，就像Rennie等说描述的那样（2003年）（本书中所用的TF × IDF和长度正常化转换可用*StringToWordVector*过滤器来实现）。

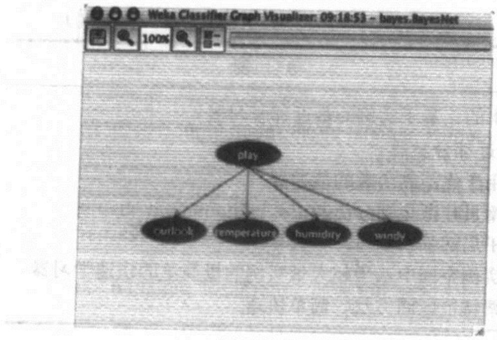
AODE平均单一依赖估计器是一种对一个交互式贝叶斯模型空间进行均分的贝叶斯方法，该方法比朴素贝叶斯（Webb等，待出版）有较弱的独立假定。该算法对含有非独立属性的数据集进行分类比朴素贝叶斯更精确。

*BayesNet*通过基于6.7节中所做的假设来学习贝叶斯网络，必须是名词性属性（数值性属性需预先离散）及无残缺值（任何类似的值全部被替换）。有两种不同的用于估计网络的条件概率表的算法。搜索是通过以下方式完成的：K2或TAN算法（第6.7节），或更成熟的基于登山、模拟淬火、制表搜索及通用算法基础上的方法。作为一个可选项，搜索速度可通过AD树（第6.7节）得到提高。还有一种算法是通过条件独立测试来学习网络的结构；另外，网络结构也可从一个XML文件载入。关于Weka中贝叶斯网络的更多详细内容可在Bouckaert（2004年）中找到。

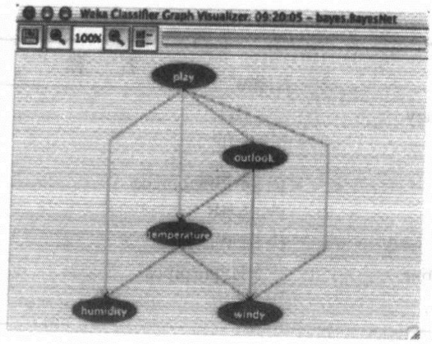
用户可通过右键单击历史条目然后选择*Visualize graph*的方式观察网络结构。图10-18a显示的是天气数据的名词性版本的图表，该图表实际上与由类值决定的所有概率条件的朴素贝叶斯的结果相对应。这是因为搜索算法在默认情况下使用的是K2，且一个节点的母节点数量的最大值设为1。通过单击配置面板上的K2将该最大值重新设为3，可得到如图10-18b所示的更为有趣的网络。在一个节点上单击可显示它的概率分布，图10-18c就是通过单击图10-18b中的*windy*节点得到的。 403～405

10.4.2 树

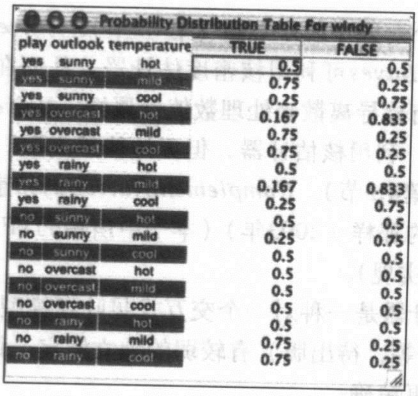
从表10-5中的树分类器中我们已经看到如何使用实现了C4.5（第6.1节）的J4.8。要显示它的选项，单击图10-4b中的Choose按钮旁边的横栏，弹出图10-19中的对象编辑器。用户可建立二叉树而不是多叉树。用户还可设定修剪的置信度阈（默认是0.25），和一个叶节点上可允许的实例的最低数量（默认是2）。用户还可选择减少误差的修剪（第6.2节）作为标准C4.5



a) 默认输出



b) 在搜索算法中将父母节点数量的最大值设为3


Probability Distribution Table For windy

play	outlook	temperature	TRUE	FALSE
yes	sunny	hot	0.5	0.5
yes	sunny	mild	0.75	0.25
yes	sunny	cool	0.25	0.75
yes	overcast	hot	0.167	0.833
yes	overcast	mild	0.75	0.25
yes	overcast	cool	0.75	0.25
yes	rainy	hot	0.5	0.5
yes	rainy	mild	0.167	0.833
yes	rainy	cool	0.25	0.75
no	sunny	hot	0.5	0.5
no	sunny	mild	0.25	0.75
no	sunny	cool	0.5	0.5
no	overcast	hot	0.5	0.5
no	overcast	mild	0.5	0.5
no	overcast	cool	0.5	0.5
no	rainy	hot	0.5	0.5
no	rainy	mild	0.75	0.25
no	rainy	cool	0.75	0.25

c) 图b)中*windy*节点的概率分布表

406 图10-18　可视化一个基于天气数据（名词性版本）的贝叶斯网络

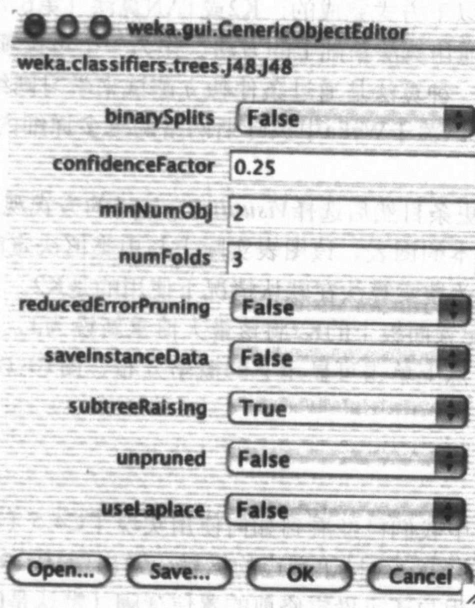


图10-19　改变J4.8的参数

修剪的替补。参数*numFolds*（默认是3）决定了修剪集的大小，数据按照折的数量被平均划分，最后一折用作修剪。当可视化所得的树时（10.1节），最好能够参照原始数据点，用户可通过打开*saveInstanceData*做到这一点（默认是关掉或*False*，目的是节省存储器）。用户可阻止子树上升，从而得到一个更高效的算法；强迫所用算法使用未修剪的树而不是已修剪的树；或将拉普拉斯平滑应用于预测的概率（第4.2节）。

表10-5列出了很多其他决策树方法。*Id3*是第4章中解释过的基本算法。*DecisionStump*是专门设计用来与后面将要讨论的提升方法一起使用的，它为含有一个类别化的，或数值性的类的数据集建立一个单层二叉决策树，与此同时，它把残缺值按照一个单独的值来处理并从树桩上扩展出第三个分支。通过*RandomTree*所构建的树测试每个节点上的指定数量的随机属性，对树不做修剪。*RandomForest*通过将随机树进行整体装袋来构建随机森林（第7.5节）。

*REPTree*用信息增益/方差缩减生成一个决策树或回归树并用减少－误差的修剪法对其进行修剪（第6.2节）。为了在速度上进行优化，该算法只对数值性属性的值进行排序一次（第 407
6.1节）。它像C4.5一样把实例分解成碎片来处理残缺值。用户可设定每个叶所含实例的最低数量，树的最大深度（当提升树时很有用），分割训练集时方差的最小比例（只针对数值性的类），和修剪时折的数量。

*NBTree*是决策树和朴素贝叶斯的混合产物。它所创建的树的叶可用作所有到达该叶的实例的朴素贝叶斯分类器。在构建树时，交叉验证被用于决定一个节点是否应当进一步被分割，还是改用朴素贝叶斯模型（Kohavi，1996年）。

*M5P*是第6.5节中描述过的模型树学习器。*LMT*可用于建立logistic模型树（第7.5节）。*LMT*还可处理二类及多类值目标变量，数值性和名词性属性及残缺值。当在一个节点上拟合一个logistic回归函数时，通过交叉验证来决定一次运行多少次循环，然后将该运行次数应用到树的所有其他节点，而不是在每个节点都进行交叉验证。这种探索方式（用户可将此方式关闭）只对精确度稍有影响，但却在相当程度上缩短了运行时间。另外，用户还可设定提升循环的次数，该次数应用于树的每个节点。通常情况下，使用交叉验证会使由误分类导致的失误减至最低，但也可以选择使用概率的根均方差。分割的依据可基于C4.5的信息增益（默认设置）或logitBoost的残留部分，目的是力求提高残留部分的纯度。

*ADTree*用提升方法生成一个交互式决策树（第7.5节），且优化成二类问题。提升循环的次数只是其中的一个参数，且该参数可被调节，以便适应数据集及在复杂程度和精确度之间进行任意折衷。除非发生节点的合并，否则每次循环会在树上加入三个节点（一个分割节点和两个预测节点）。默认的搜索方法是穷举搜索（引申至所有路径）；其余的方法则是探索式的，并且要快很多。用户可决定是否存储实例数据以用作可视化。

10.4.3 规则

表10-5列出了许多生成规则的方法。*DecisionTable*生成一个决策表多数分类器（第7.1节）。它使用最佳－优先探索评估属性子集，并且可用交叉验证进行评估（Kohavi，1995年）。其中一个选项使用最近邻方法并基于同样的属性集来决定每个未被决策表中所涵盖的实例的类属性，这里的涵盖指的是实例必须与决策表中的某个具体条目相吻合，而不是仅仅与该表的全局多数相一致。*OneR*是只含有如下一个参数的1R分类器（第4.1节）：最小的离散桶的大小。*ConjunctiveRule*只学习单一的规则用以预测要么是数值性要么是名词性的类值。未被规则覆 408

盖的测试实例被指定为未被覆盖的训练实例的默认类值（或分布）。先计算出每个前提条件的信息增益（名词性的类）或方差缩减（数值性的类），然后用减小误差的修剪对规则进行修剪。*ZeroR*更为简单，只需预测测试数据的多数类（如果是名词性的）或平均值（如果是数值性的）。*Prism*实现规则的初级覆盖算法（第4.4节）。

*Part*由部分决策树中获取规则（第6.2节）。它使用与J4.8中同样的用户自定义的参数，利用C4.5的探索方法来生成树。*M5Rules*利用M5'生成的模型树获取回归规则（第6.5节）。*Ridor*从产生默认规则的特例中学习规则（第6.2节），它用递增减少－误差修剪法来找出具有最小误差率的特例，从而为每个特例找到最好的特例，并依此进行循环。

*JRip*实现RIPPER（第6.2节），包括规则集的探索式全局优化（Cohen，1995年）。*Nnge*是一种利用非嵌套的泛化的样本集来产生规则的最近邻方法（第6.4节）。

10.4.4　函数

表10-5中的函数类别包括一组多种类的分类器，这些分类器可以以一种自然合理的方式写成数学方程式。而像决策树和规则等其他的分类器则不可以（当然也有例外：朴素贝叶斯就有一个简单的数学公式）。这些函数中有三种实现是线性回归（第4.6节）。*SimpleLinearRegression*基于一个单独的属性学习一个线性回归模型，它会挑选能产生最小方差的那个属性。残缺值和非数值性属性在挑选时不予考虑。*LinearRegression*所做的是标准的最小平方线性回归，且可以选择性地进行属性选择，选择方式要么使用反向消除（7.1节）逐一选择，要么用全部属性建立一个完整模型，将这些属性按它们的标准化系数进行降序排列，然后逐一去掉每个项直至满足终止条件（这个方法在第6.5节中修剪树标题下曾做过描述，当时的情况和现在略有不同）。两种选择方式用的都是第6.7节中的AIC终止条件的其中一个版本。实现中含有两种进一步的细分：一种探测共线属性的机制（可被关掉），和一个能够使恶化的案例得以稳定并通过加上较大的惩罚系数降低过度拟合的*ridge*参数。从技术角度讲，*LinearRegression*实现岭回归，这在标准的统计教科书中有专门的描述。

*LeastMedSq*是一种能够使以回归线为基准的发散的平方的中值（而不是平均值）最小化
409 的高可靠性线性回归方法（第7.4节）（Rousseeuw和Leroy，1987年）。它将标准线性回归反复应用于数据的子采样，然后把具有最小中值平方误差的解决方法显示出来。

*SMO*采用多项式或高斯核（Platt，1998年；Keerthi等，2001年）来实现连续最小优化算法用于训练一个支持向量分类器（第6.3节）。数据中的残缺值全部被替换；名词性属性转换成二进制属性；且默认条件下属性须进行正常化，要注意，输出结果中的系数是基于经过正常化的数据的基础上。用户可将正常化功能关闭，或将输入的值标准化至零平均值及单位方差。成对分类多用于多类的问题。logistic回归模型可与支持向量机相吻合以获取概率评估。对于多类的情形，所预测的概率则是对偶关联的（Hastie和Tibshirani，1998年）。在处理稀疏实例时，将正常化功能关闭可加快处理速度。*SMOreg*实现连续最小优化算法用于处理回归问题（Smola和Schölkopf，1998年）。

*VotedPerceptron*是投票感知器算法（第6.3节）。*Winnow*（第4.6节）修改基本的感知器以便利用倍增的更新。该实现允许有不同于$1/\alpha$ 的第二个倍数β以便用于图4-11中的除法运算，并且还提供所用算法的平衡版本。

*PaceRegression*用新的Pace回归技术（Wang和Witten，2002年）建立线性回归模型。在有

很多属性的情况下，Pace回归特别擅长决定哪些属性应该被删除，实际上，在某些常见情况下，当属性的数量趋于无限大时，Pace回归可被证明是最佳选择。

*SimpleLogistic*建立logistic回归模型（第4.6节），使用以简单回归函数为基学习器的LogitBoost来拟合这些模型并用交叉验证来决定进行多少次循环，因为交叉验证可支持自动属性选择（Landwehr等，2003年）。*Logistic*是另外一种建立在le Cessie和van Houwelingen（1992年）研究成果基础上的、用于生成和利用一个多名词性的logistic回归模型及一个岭估计器的实现方法，该方法通过为回归项加上较大的惩罚系数来防止过度拟合。

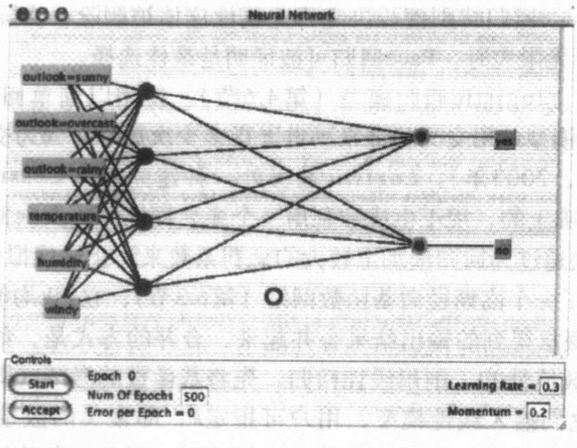
*RBFNetwork*实现了一个高斯径向基函数网络（第6.3节），它用k均值推导出隐藏项的中心点及宽度，并将由隐藏层得到的输出结果合并起来。合并的方式是，如果类是名词性的就用logistic回归；如果是数值性的，则用线性回归。先将基函数的激活（即输出）正常化，即相加总和为1，然后将它们输入线性模型。用户可指定k，即聚类的数量；名词性类的问题的logistic回归循环次数的最大值；相关聚类的最小标准差；以及回归的岭值。如果类是名词性的，则将k均值分别应用于每个类，从而为每个类导出k个聚类。 410

神经网络

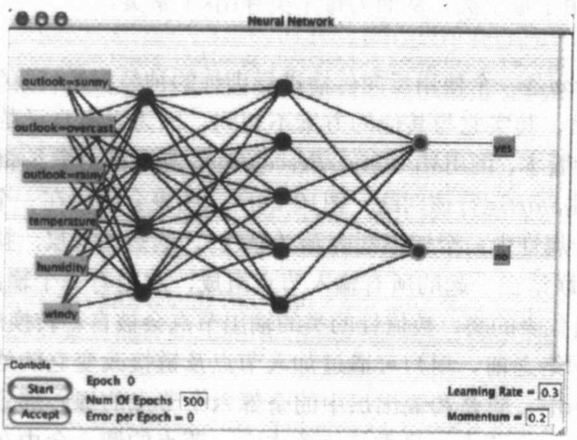
*MultilayerPerceptron*是一个使用反向传播进行训练的神经网络（第6.3节）。尽管在表10-5中它是列在函数的下面，其实它与其他的方案不相同，因为它有自己的用户界面。如果用户载入天气数据的数值性版本，调用*MultilayerPerceptron*，在它的对象编辑器中将*GUI*设为*True*，并在*Classify*面板上单击*Start*运行该网络，图10-20所示图表会出现在一个单独的窗口中。该网络有三层：在左侧每个属性由一个矩形框表示（绿色），这是输入层；接下来在它的旁边有一个隐蔽层（红色），由联结在一起的所有输入节点组成；右边有一个输出层（橙色）。最右边的标签表明输出节点所代表的类。数值性的类的输出节点会被自动转换成无阈值的线性单元。

在单击*Start*运行网络之前，用户可通过加入节点及链接改变它的结构。节点可被选定或取消选定。在图10-20a中，隐蔽和输出层中的全部六个节点的圆心部分都呈现灰色，表明它们都未被选定。要选定一个节点，只需简单单击它，节点的圆心会由灰色变为亮黄色。要取消选定一个节点，在空白处右键单击。要加入一个节点，首先确保没有节点被选定，然后在面板上的任意处左键单击；新节点将自动选定。图10-20a中，在下方的中间处，一个新节点已经被加入。要将两个节点连接起来，选定起始节点，然后单击终止节点。如果有好几个起始节点被选定，它们都会被连接到终止节点。如果用户在空白处单击，一个新节点会生成并作为终止节点。请注意，连接是有方向的（尽管它们的方向未被显示出来）。起始节点会保持被选定状态；这样做的好处是，用户只需点击几下鼠标即可加入整个隐蔽层如图10-20b。要删除一个节点，首先确保没有节点被选定，然后右键单击要删除的节点；这样做所删除的不只是该节点，所有指向该节点的连接也一并删除。要删除一个单独的连接，选定一个节点，右键单击该连接另外一端的节点。

除了配置网络的结构，用户还可控制学习比率和动量（第6.3节），以及该网络穿越数据的总的通过次数，称为纪元（epochs）。用户单击*Start*时，网络即开始训练，且该纪元正常运行的指示及纪元的误差会显示在图10-20中面板的左下方。请留意，误差是基于整个网络基础上的，随着误差值的计算，网络也随之变化。对于数值性的类来说，误差值取决于类是否被正常化。当时间点的数量达到指定值时，网络会停止，此时，用户可选择接受该结果或将纪元的数量增至所期望的值，再点击*Start*继续训练。



a)



b)

图10-20　使用Weka的神经网络图形用户界面

*MultilayerPerceptron*无需通过图形界面也可运行。通过对象编辑器设置几个参数即可控制它的操作。如果用户使用图形界面，则可对网络起始结构进行管理，即用户可互动式地对结构进行重新设定。使用*autoBuild*即可加入隐蔽层并将其连接起来。默认条件下隐蔽层是不
411~412 显示的，如图10-20a所示，但如果不用*autoBuild*，该层是无法显示出来的，并且也没有连接。参数*hiddenLayers*定义了哪些隐蔽层会被显示及每个隐蔽层含多少节点。图10-20a是在值的大小为4的情况下产生的（每个隐蔽层含4个节点），尽管图10-20b是通过交互式加入节点生成的，它也可同样通过将*hiddenLayers*设为4，5（一个隐蔽层有4个节点，另一个有5个节点）来产生。所设的值是一串由逗号分隔的整数；0表示没有隐蔽层。进一步说，如果不用整数值，则可以使用一些预设的值：*i*是属性的数量，*o*是类值的数量，*a*是二者的平均值，而*t*表示它们的和。*a*是默认值，图10-20a就是用它产生的。

参数*learningRate*和*Momentum*为这些变量设定值，它们也可通过图形界面进行重设。参数*decay*会导致学习比率随着时间的增加而下降：它用比率的起始值除以周期的数量来得到当前的比率。该方法有时可提升性能，且可防止网络发散。参数*reset*会自动在网络偏离答案时

重新为网络设定一个较低的学习比率并重新开始训练（这个选项仅在不使用图形用户界面时才有效）。

参数*trainingTime*用于设定训练纪元的次数。除此之外，还可以提取一定百分比的数据专门用于确认（使用*validationSetSize*）：然后训练继续进行直至其在验证集上的性能开始持续恶化，或直至达到所指定的纪元的次数。如果该百分比设为零，则不使用验证集。参数*validationThreshold*决定了验证集的误差可允许持续恶化多少次才停止训练。

在*MultilayerPerceptron*对象编辑器中默认情况下所指定的过滤器是*nominal ToBinary Filter*；如果所用数据的名词性属性的确是按照序数排列的话，将该过滤器关闭则可对提高性能有帮助。属性可被正常化（用*normalizeAttributes*），一个数值性的类属性也可以正常化（用*normalizeNumericClass*）：两种方式都能提高性能。

10.4.5 懒惰分类器

懒惰分类器把训练实例存储起来直到进行分类时才开始真正工作。*IB1*是一个基本的基于实例的学习器（4.7节），它能找出与给定的测试实例在欧几里得距离上最接近的训练实例，然后将该测试实例预测为与训练实例同样的类。如果有不止一个的训练实例都被认为是最接近的，使用第一个发现的实例。*IBk*同样是一个以距离为度量的*k*最近邻分类器。最近邻的数量（默认*k*=1）可在对象编辑器中明确指定或用留一交叉验证自动确定，这取决于所指定的值的上限。从一个以上的邻居中做出的预测可按照这些邻居与测试实例的距离进行加权，有两个不同的方程式被实现以用于将距离转换为权。分类器持有的训练实例的数量可通过设定窗口尺寸选项来加以限制。随着新的训练实例不断加入，最早的实例被删除以使训练实例保持为所限制的数量。*KStar*是一个使用泛化距离函数的最近邻方法，该函数是基于转换基础上的（第6.4节）。 413

LBR（懒惰贝叶斯规则的缩写）是一个贝叶斯分类器，它将所有的处理过程都推迟到分类时进行。该分类器从每个测试实例的属性中挑选出一组不可以对其进行独立假设的属性；除了给定类属性和所选定的属性以外的其他属性则可被看作相互独立。该分类器对于小型测试集来说效果很好（Zheng和Webb，2000年）。

*LWL*是一个用于局部加权学习的通用算法。它用一个基于实例的方法来进行权指定，并根据经过加权的实例生成一个分类器。该分类器是在*LWL*的对象编辑器中选定的：一个较好的选择是分类问题用朴素贝叶斯，而回归问题则用线性回归（第6.5节）。用户可设定所用的邻居数量、决定核的带宽，以及用于加权的核的形状是线性、反转或高斯。属性正常化功能默认情况下是打开的。

10.4.6 其他的杂项分类器

Misc.类别包含两个简单的在第4.7节提到过的分类器。对于离散的分类问题来说，*HyperPipes*记录了在训练数据中观察到的每个属性的值的区间及其类别，并得出哪些区间含有测试实例的属性值，从而挑出含有最多正确区间的类别。*VFI*（投票特征区间）通过离散数值性属性及对名词性属性使用点区间的方式围绕每个类建立区间，并且针对每个属性记录每个区间中类的数量，然后用投票的方式对测试实例进行分类（Demiroz和Guvenir，1997年）。一个简单的属性加权方案会为较多的置信度区间指定较高的权，这里置信度是熵的函数。*VFI*

比朴素贝叶斯速度快，但比*HyperPipes*慢。无论哪种方法都无法处理残缺值。

10.5 元学习算法

表10-6中列出的元学习算法将分类器变成功能更强大的学习器。其参数之一用来指定基分类器；其余参数则用来确定类似装袋和提升方案中循环的次数，以及随机数生成器的初始种子。我们在第10.3节中已经见过*Filtered Classifier*，它在经过过滤器过滤的数据上运行一个分类器，所用的过滤器是一个参数。过滤器本身的参数则是专门建立在训练数据基础上的，这种方法对于将有指导过滤器应用于测试数据很合适。

10.5.1 装袋和随机化

*Bagging*将一个分类器袋装起来以减小方差（第7.5节）。这种实现方法对分类和回归都很有效，当然这取决于基学习器。对于分类来说，预测是通过平均概率估计而不是投票做出的。
414 其中一个参数是袋的尺寸，它是用训练集的百分比表示的。另一个参数是是否计算溢袋误差（out-of-bagerror），也就是全体成员的平均误差（Breiman，2001年）。

*RandomCommittee*更加简单，它生成一个基分类器的合集，然后平均它们的预测。每个分类器都是基于同样的数据但用不同的随机种子（第7.5节）。这种方法只有在基分类器被随机化以后才有意义，否则所有的分类器都会一模一样。

10.5.2 提升

*AdaBoostM1*实现的是第7.5节中描述过的算法（图7-7）。它可通过指定一个用于权修剪的阈来加速。如果基分类器无法处理经过加权的实例，*AdaBoostM1*就会进行重采样（用户当然也可强迫它进行重采样）。*MultiBoostAB*将提升与一个装袋的变体合并在一起，以防止过度拟合（Webb，2000年）。

与提升只适用于名词性类不同，*AdditiveRegression*可强化一个回归学习器的性能（第7.5节）。它有两个参数：收缩，用来管理学习比率，还有一个是产生模型数量的最大值。如果后一个参数是无穷大的话，运行会持续下去直至误差不再降低。

表10-6 Weka中的元学习算法

名称		功能
Meta	AdaBoostM1	用AdaBoostM1方法提升
	AdditiveRegression	通过迭代拟合残留部分来提高回归方法的性能
	AttributeSelectedClassifier	通过属性选择来降低数据的维数
	Bagging	袋装一个分类器；对回归问题也同样有效
	ClassificationViaRegression	用回归方法进行分类
	CostSensitiveClassifier	使其基分类器对成本敏感
	CVParameterSelection	通过交叉验证进行参数选择
	Decorate	通过使用特别构建的人工训练范例来生成分类器合集
	FilteredClassifier	在经过过滤的数据上运行一个分类器
	Grading	以已经被标记为正确或不正确的底层预测作为输入的元学习器
	LogitBoost	进行相加的logistic回归
	MetaCost	使一个分类器成为成本敏感

（续）

名　称	功　能
MultiBoostAB	用多重提升方法将提升和装袋合并
MultiClassClassifier	将二类分类器用于多类数据集
MultiScheme	使用交叉验证从多个候选对象中选择一个分类器
OrdinalClassClassifier	将标准分类算法应用于具有序数性类值的分类问题
RacedIncrementalLogitBoost	通过使logitboosted committee相互间竞赛来进行以批次为基础的递增学习
RandomCommittee	生成一个可被随机化的基分类器的合集
RegressionByDiscretization	离散类属性并使用一个分类器
Stacking	用堆栈的方法将多个分类器组合起来
StackingC	堆栈的更高效版本
ThresholdSelector	优化一个F测量用于概率分类器
Vote	使用概率估计或数值预测平均值来合并分类器

*Decorate*通过使用特别构建的人工训练范例生成一个各种各样分类器的合集。该技术据说可通过基分类器、通过装袋和随机森林元学习器持续地改进（Melville和Mooney，2005年）[⊖]。对于小型训练集来说，它的性能比提升要好，且在大型数据集上也不算差。它的参数之一是所用人工范例的数量，它是用训练数据的比例表示的。另一个参数是合集中欲加入的分类器的数量，因为循环次数的上限也可人为设定，运行过程也许会过早地停下来。较大的合集通常会生成更精确的模型，但也会增加训练时间及模型的复杂程度。

*LogitBoost*进行的是相加的logistic回归（第7.5节）。像*AdaBoostM1*一样，它也可通过一个用于加权修剪的阈来加速。合适的迭代次数可通过内部交叉验证得出；调节收缩参数可防止过度拟合；且用户还可选择重采样而不是重加权。*RacedIncrementalLogitBoost*通过令LogitBoostedCommittee之间互相竞赛来学习，且通过将数据分批次处理(8.1节)来进行递增操作，因此对处理大型数据集很有用（Frank等，2002年）。每个委员会成员都由不同的批次得出。批次的规模先由一个给定的最小值开始，然后反复成倍递增直至达到一个预设的最大值。如果基分类器无法处理经过加权的实例，就必须重新采样（用户当然也可强迫进行重采样）。对数似然修剪可用于每个委员会中：这样，如果新的委员会成员降低验证数据的对数似然，它们就会被删除。用户可决定将多少个实例提取出来专门用于验证。当训练结束时，用于验证的数据还用于决定哪个委员会保留下来。 415~416

10.5.3 合并分类器

*Vote*提供了一个基准方法用于合并分类器，合并是通过将分类器的概率估计（分类）或数值预测（回归）平均的方式完成的。*MultiScheme*利用百分比精确度（分类）或均方差（回归）的交叉验证，从一组候选分类器中选择最佳的一个。折的数量是一个参数。在训练数据上的性能表现也可作替代使用。

*Stacking*利用堆栈（第7.5节）将分类器合并起来，这对于分类及回归问题都适用。用户可指明基分类器，元学习器以及交叉验证的折的数量。*StackingC*实现一个效率更高的变体，

⊖ 随机森林方案在10.4节中提到过。它实际上是一个元学习器，但Weka之所以将它与其他决策树方法一起包括进来，是因为它与一个具体的分类器*RandomTree*紧密相关。

它要求其元学习器必须是一个数值性的预测方案（Seewald，2002年）。在*Grading*中，元学习器的输入必须是已经被标记（即经过评分的）为正确或不正确的底层预测。对应着每个基分类器都会得出一个元学习器。以便当基分类器出错时，元学习器用作预测。就像堆栈可以被看作是投票的泛化，评分通过交叉验证泛化了选择过程（Seewald和Fürnkranz，2001年）。

10.5.4 成本敏感学习

有两种用于成本敏感学习的元学习器（第5.7节）。成本矩阵可作为一个参数提供或由文件载入，该文件来自于由*onDemandDirectory*属性设定的文件夹中，并且是用关系名字加上扩展*.cost*命名的。*CostSensitiveClassifier*要么根据指定给每个类的总成本对训练实例进行重加权（成本敏感学习，5.7节），或者用最小而不是最可能的预期错误分类成本来预测类（成本敏感分类，5.7节）。*MetaCost*从基学习器中产生一个成本敏感分类器（第7.5节）。该实现方法在对训练数据重新进行分类时使用全部的装袋循环。（Domingos，1999年在仅仅使用那些含有每个训练实例的循环来做重新分类时观察到了些许改进）。用户可以指定每个袋的尺寸及装袋的循环的次数。

10.5.5 优化性能

三种元学习器使用包装技术来优化基分类器的性能。*AttributeSelectedClassifier*选择属性，降低数据的维数，然后将其传给分类器（第7.1节）。用户可利用第10.2节中描述过的*Select attributes*面板来挑选属性评估器和搜索方法。*CVParameterSelection*通过使用交叉验证选择参数的方式来优化性能。用户可为每个参数提供一个含有下限和上限以及想要的递增量的一个字符。例如，要想使参数 –*P*从1到10以每次1的方式递增，用P 1 10 1 。交叉验证的折的数量
417 也可以指定。

第三个元学习器*ThresholdSelector*通过为分类器的输出选定一个概率阈来优化F测量（第5.7节）。其性能可通过训练数据，一个旁置集来衡量，或通过交叉验证。基学习器返回的概率可按比例转换为[0，1]的区间，如果方案的概率被限定在一个狭窄的子区间，这么做很有帮助。元学习器可通过指定类值的方式应用于多类问题，进行优化时所指定的类值有

1) 第一个类值
2) 第二个类值
3) 最不频繁的类值
4) 最频繁的类值
5) 名字为yes，pos（itive）或1的第一个类属性

10.5.6 针对不同任务重新调整分类器

四种元学习器可以把为完成一种任务而设计的学习器进行改装，从而可完成其他任务。*ClassificationViaRegression*利用回归方法，通过使类属性二元化及，为每个值建立一个回归模型的方式进行分类。*RegressionByDiscretization*是一种回归方案，它用等宽离散法将类属性离散成给定数量的箱，然后再使用一个分类器。预测结果就是每个离散区间的中间的类值的加权平均值，其中所用的权基于每个区间的预测概率。*OrdinalClassClassifier*将标准的分类算法应用于有序类的问题（Frank和Hall，2001年）。*MultiClassClassifier*将二类分类器用于多类问题，使用以下方法中的任意一种:

1) 其中之一对应所有其他
2) 使用投票作预测的成对分类
3) 穷举纠错代码（第7.5节）
4) 随机选取的纠错代码

随机代码向量已知具有好的纠错属性，它的一个参数明确了代码向量的长度（以位为单位）。

10.6 聚类算法

表10-7列出了Weka的聚类算法；前两种算法以及*SimpleKMeans*在第6.6节中做了描述。对于*EM*实现，用户可指定需要产生多少聚类，否则所用的算法可通过交叉验证来决定，在这种情况下，折的数量固定为10（除非训练实例少于10个）。用户可指定循环次数的最大值，并且为正常的密度计算设定可允许的最小标准差。*SimpleKMeans*使用*k*均值来聚类数据；聚类的数量通过一个参数指定。*Cobweb*实现了用于名词性属性的Cobweb算法和用于数值性属性的Classit算法。对于合并和分割运算符的排序及优先权来说，原始的Cobweb与Classit报告中不尽相同（这里多少有些歧义）。该实现总是先比较四种不同的处理新实例的方式，然后选择其中最好的：将其与最佳主体节点（host）相加，使其成为一个新的叶，将两个最佳主体节点合并后再将其加入合并后的节点，分割最佳主体节点然后将其加入分割后的节点之一。*Acuity*和*cutoff*是参数。 418

表10-7 聚类算法

名称	功能
EM	使用期望最大化进行聚类
Cobweb	实现Cobweb和Classit聚类算法
FarthestFirst	使用由远端开始遍历(farthest first traversal）算法进行聚类
MakeDensityBasedCluster	将一个聚类器包装，使其返回分布和密度
SimpleKMeans	使用*k*均值方法进行聚类

*FarthestFirst*实现Hochbaum和Shmoys（1985年）远端优先遍历算法，Sanjoy Dasgupta（2002年）曾做过引述；它是一个快速，简单，以*k*均值为模型的近似的聚类器。*MakeDensityBasedCluster*是一个元聚类器，它包装一个聚类算法，使其返回一个概率分布和密度。它为每个聚类拟合一个离散分布，或一个对称的正态分布（其最小标准差是一个参数）。

10.7 关联规则学习器

Weka有三种关联规则学习器，如表10-8中所列。*Apriori*实现Apriori算法（第4.5节）。它由数据项的最少支持度100%开始，逐步递减5%，直到至少有所要求的最小置信度为0.9的10个规则，或者支持度达到了10%的较低极限，二者以先出现为计。(这些默认值是可被修改的。)共有四种不同的因素用于决定排序规则：置信度，即同时被前提条件和结论所涵盖的范例比例（在第4.5节中称为精确度）；提升度，即等于置信度除以支持度（在第4.5节中称为涵盖度）；平衡度，在前提条件和结论是统计独立的条件下，被前提条件和结论所同时涵盖的超出期望值的那部分范例的比例；可信度，一种由Brin等（1997年）确立的测量法。用户可以指一个显著性级别，规则将会在这个级别上进行显著性测试。

表10-8　关联规则学习器

名　称	功　能
Apriori	用Apriori算法寻找关联规则
PredictiveApriori	找出经过预测精度排序的关联规则
Terius	确认指引下的关联或分类规则的发现

*PredictiveApriori*将置信度和支持度合并为预测精度而成为单一度测量法（Sheffer，2001年），然后依次寻找*n*个最好关联规则。对算法本身来说，由于预测精度取决于支持度阈，该算法不断地提高该阈值。*Tertius*根据确认度来寻找规则（Flach和Lachiche，1999年），它像Apriori一样寻找其结论中含有多重条件的规则，但不同的是，这些条件相互间是‘或’，而不是‘与’的关系。它可被设定为寻找只预测一个单独条件或其他事先预定的属性的那些规则
418 (例如，分类规则)。可由一个参数来决定是否允许否定条件在前提或结论，或两者中同时出现；其他参数控制着寻找规则的数量，最低确认度，最低涵盖度，反向实例（counter-instances）的最高比例，规则的最大范围。残缺值可以匹配任意值，或干脆不匹配，或可起显著作用且可能出现在规则中。

10.8　属性选择

图10-21展示了Weka的属性选择面板的局部，用户可在上面设定属性评估器和搜索方法；表10-9和表10-10列出了所有选项。属性选择一般通过搜索并评估属性子集空间来实现（第7.1节）。这可通过结合表10-9中的四个属性子集评估器的其中之一和表10-10中七种搜索方法的其中之一来完成。一个具有快速运行潜力但不是很精确的方法是，对属性逐一评估然后进行排列，删除截止点以下的所有属性。这可通过挑选表10-9中八个单一属性评估器的其中之一，再使用表10-10中的排序法来完成。Weka界面允许用户从表10-9中选定一种选择方法，也可以从表10-10中选定一种搜索方法，如果用户选择了不适当的组合，Weka界面会反馈出错信息。用户可根据状态栏中的指示到出错日志中查看信息（参看第10.1节的结尾部分）。

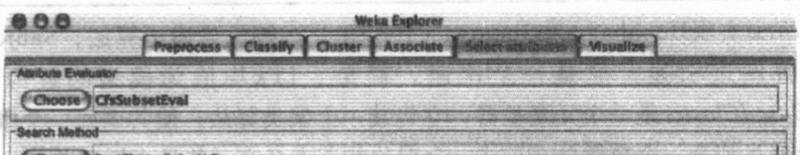


图10-21　属性选择：设定属性评估器和搜索方法

10.8.1　属性子集评估器

子集评估器取一个属性子集，返回指导搜索的一个数值性信息。它们的配置方法与Weka的其他对象一样。*CfsSubsetEval*逐一评估每个属性的预测能力和它们之间的重复程度，然后挑选那些与类有高度关联但相互之间关联程度却较低的属性（第7.1节）。当属性集中没有一个属性和所要加入的属性的关联程度更高时，有一个选项可以令评估器循环地加入与类有最高关联度的属性。残缺值视为一个单独的值，或者根据出现频率的比例将它的出现次数分摊到其他值中。*ConsistencySubsetEval*在训练数据映像到属性集上时，以类值的一致性来评估属

性。由于任何一个属性子集的一致性不可能比整集的好，因此，这个评估器通常和用于寻找与整集一致性相等的最小属性子集的随机或穷举搜索法一起使用。

上面所提到的子集评估器实际上是属性选择的过滤器方法（第7.1节），下面将要介绍包装方法。*ClassifierSubsetEval*使用一个在对象编辑器中作为参数指定的分类器来评估训练数据中的属性集，或一个单独旁置集中的属性集。*WrapperSubsetEval*也采用分类器评估属性集，但它应用交叉验证估计学习方案在每个属性集上的正确率。

表10-9 用于属性选择的属性评估方法

	名　称	功　能
属性子集评估器	CfsSubsetEval	综合考虑单一属性的预测值和属性间的重复度
	ClassifierSubsetEval	用分类器评估属性集
	ConsistencySubsetEval	将训练数据集映射到属性集上来检测类值的一致性
	WrapperSubsetEval	使用分类器和交叉验证
单一属性评估器	ChiSquaredAttributeEval	计算每个属性的基于类的χ^2统计数据
	GainRatioAttributeEval	以增益比率为依据的属性评估
	InfoGainAttributeEval	以信息增益为依据的属性评估
	OneRAttributeEval	用OneR的方法论来评估属性
	PrincipalComponents	进行重要组件的分析和转换
	ReliefFAttributeEval	基于实例的属性评估器
	SVMAttributeEval	使用线性支持向量机来决定属性值
	SymmetricalUncertAttributeEval	以对称不确定性为依据的属性评估

表10-10 用于属性选择的搜索方法

	名　称	功　能
搜索方法	BestFirst	具有返回的贪心登山式搜索
	ExhaustiveSearch	穷举搜索
	GeneticSearch	使用一个简单遗传的算法进行搜索
	GreedyStepwise	不具有返回的贪心登山式搜索；具有产生属性排序列表的可选项
	RaceSearch	使用竞赛搜索方法
	RandomSearch	随机搜索
	RankSearch	排列属性并使用属性子集评估器将有潜力的属性进行排序
排序方法	Ranker	按照属性的评估对它们进行逐一（而不是按子集）排序

10.8.2 单一属性评估器

单一属性评估器与*Ranker*搜索方法一起可用于产生一个有序列表，*Ranker*可从列表中删除一个给定的数字（后面会有解释）。这些评估器也可用于*RankSearch*方法。*ReliefFAttributeEval*是基于实例的，它对实例进行随机采样，并检查相邻的具有相同的及不同的类的实例（第7.1节）。该评估器运行于离散的及连续的类的数据。其参数可指明待采样实例数，待检查的邻居的数量，是否依据距离对邻居进行加权，以及一个用于控制权伴随距离所产生衰减的剧烈程度的指数函数。

*InfoGainAttributeEval*通过测量与类相关的信息增益来评估属性。它首先用基于MDL的离

散方法离散数值性属性（也可设定为将属性二元化而不是离散）。该方法，加上下面要谈到的
419～422 三种方法，可将*missing*作为一个单独的值来处理，或者将残缺值的数量与其他值一起按照与它们的频率成正比进行分布。*ChiSquaredAttributeEval*通过计算与类相关的χ^2统计来评估属性。*GainRatioAttributeEval*评估属性的方式是通过测量这些属性的与类值相关的增益比率。*SymmetricalUncertAttributeEval*通过测量属性A与类C相关的对称不确定性进行评估（第7.1节）。

*OneRAttributeEval*使用OneR分类器所采用的简单的精度测量方法。它可以像OneR一样用训练数据作评估，或应用内部交叉验证，折的数量是一个参数。它采用OneR的简单离散方法，最小离散桶是一个参数。

*SVMAttributeEval*使用递归属性排除及线性向量机作属性评估（第7.1节）。属性是根据它们的系数大小一个接一个被选定，并逐一重新学习。为加快属性选择的速度，可在每个阶段删除一定数量（或比例）的属性。实际上，可设定一个比例以确保只保留一定数量的属性，接下来改用固定数量属性的处理方法，如此即可以快速排除很多属性，然后集中更多注意力考虑剩下的每个属性。很多不同的参数传递给支持向量机：复杂度、ε、容许偏差及所用的过滤方法。

与其他单一属性评估器不同，*PrincipalComponents*转换的是整个属性集。新属性是按照它们的特征值的大小排序的（第7.3节）。一个可选项是，可通过挑选足够的特征向量以构成给定比例的方差（默认值是95%）。用户还可使用该评估器将经过缩减后的数据转换回原来的空间。

10.8.3 搜索方法

搜索方法遍历整个属性空间以期找出一个好的子集。品质好坏是通过选定的属性子集评估器来衡量的。每种搜索方法都可通过Weka的对象编辑器进行配置。*BestFirst*通过返回进行贪心式爬山搜索；用户可指定系统在连续遇到多少个未被改进的节点后才返回。它可以从一个空的属性集正向搜索，或从一个满集反向搜索，或从中间的一个点开始（由一列属性索引指定）并向前后两个方向，通过考虑所有可能的单个属性加入及删除进行搜索。已经评估过的子集将被保存在高速缓冲存储器中，以提高效率；缓冲器的大小是一个参数。

*GreedyStepwise*对属性子集空间进行贪心式搜索。与*BestFirst*类似，它也可以由一个空集
423 开始正向处理或从满集开始反向处理。与*BestFirst*不同的是，它不返回，一旦加入或删除最佳剩余属性导致评估度量下降时，它能够及时终止。在另外一种模式中，它通过由空至满（或由满至空）遍历属性空间以及记录属性被选定的顺序来为属性排序。用户可指定要保留的属性的数量或设定一个阈，低于阈则属性将被删除。

可与*ClassifierSubsetEval*一起使用的*RaceSearch*用竞赛搜索法来计算竞争属性子集的交叉验证误差（第7.1节）。第7.1节所描述的四种不同的搜索已经实现：正向选择，反向删除，模式搜索和排序竞赛。对于最后一项来说，一个单独的属性评估器（该评估器也可被指定）用于产生一个初始排序。如果使用正向选择，通过继续竞赛直至所有属性都被选定，从而产生一个有序属性列表也是可能的，排好的序列与属性被加入列表中的顺序相一致。与*GreedyStepwise*一样，用户也可指定要保留的属性的数量，或设定一个阈，低于阈则属性将被删除。

*GeneticSearch*利用一个简单的遗传算法（Goldberg，1989年）。它的参数包括人口数量，

世代的数量，交叉与突变的概率。作为出发点，用户可指定一个属性索引列表，它将成为起始人群的一个成员。进展报告是按照每一定数量的世代产生的。*RandomSearch*随机搜索属性子集空间。如果已有一个初始集，它会搜索与初始状态相比有改善（或等同），并且含有较少（或同样数量）的属性。否则，它会从一个随机点开始搜索并报告所发现的最佳子集。将所有属性置入初始集中，即得出Liu和Sectiono（1996年）的概率特征选择算法。用户可决定对多大比例的搜索空间进行探索。*ExhaustiveSearch*由空集开始搜索属性子集的全部空间，并报告所发现的最佳子集。如果已提供了一个初始集，它会从该起始点开始反向搜索并报告最小的具有较好（或等同）评估的最小的子集。

*RankSearch*首先用一个单一属性评估器对属性进行排列，然后用一个属性子集评估器对有潜力的子集进行排序。与通常做法一样，后者在图10-21顶部的框中指定，属性评估器在*RankSearch*的对象编辑器中是作为一个特性指定的。它首先用单一属性评估器对属性进行排列，然后用子集评估器来评估不断膨胀的子集，最佳属性。最佳属性加上下一个最佳属性，等等，不断汇报最好的子集。这种进程具有较低的运算复杂度：两种评估器被调用的次数与属性的数量呈线性关系。如果使用简单的单一属性评估器（例如，*GainRatioAttributeEval*），选择起来会非常快。

最后我们来描述*Ranker*，正如前面提到的，它不是一个属性子集的搜索方法，而是一个用于单个属性排序的方案。它根据属性的个体评估对这些属性进行排列，且必须与表10-9后 [424]
半部分列出的单一属性评估器中的一种联合使用，它不能与属性子集评估器一起使用。*Ranker*不只是对属性排序，它还通过删除序位较低的属性进行属性选择。用户可设定一个截止阈，阈以下的属性将被删除，或指明要保留的属性的数量。用户还可指定一些特定的属性，无论其序位高低，一律保留。 [425]

第11章 Knowledge Flow界面

通过知识流（Knowledge Flow）界面，用户从工具条中选择Weka组件，这些组件被置于设计画布上，连接成一个处理和分析数据的具有方向性的流程图。知识流界面为那些喜欢从数据是如何在系统中流动的这样的角度出发来思考问题的用户提供了探索者之外另外一种选择。它还允许将配置的设计与执行应用于数据流（stream data）的处理。探索者界面则无法做到这一点。用户可通过从图10-3a所示面板底部的选项中选择*KnowledgeFlow*来启动知识流界面。

11.1 开始着手

下面这个例子分步骤详细介绍了载入ARFF文件及使用J4.8进行交叉验证。我们会描述如何生成图11-1中所示的最终配置。首先，通过单击*DataSource*标签（顶部工具条最右侧）并从
427 工具条中选择*ARFFLoader*来生成一个数据源。鼠标的光标变成十字交叉表明用户此时应该放置组件。在画布上任意一点单击，一个ARFF载入器图标的复制件就会随之出现。要将此载入器与一个ARFF文件连接起来，右键单击载入器得到如图11-2a所示的弹出菜单。单击*Configure*得到图11-2b中的文件浏览器，用户可从中选择想要的ARFF文件。*File Format*下拉菜单令用户可以选择不同类型的数据，例如，电子数据表文件。

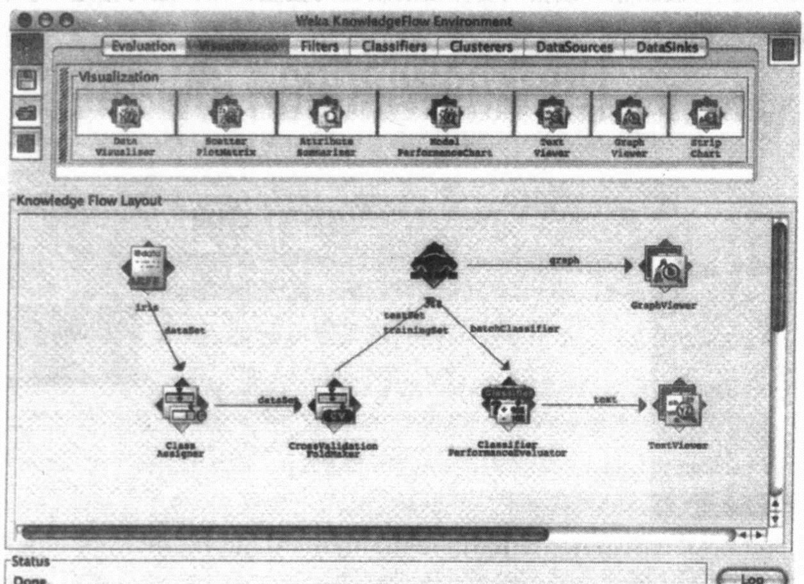
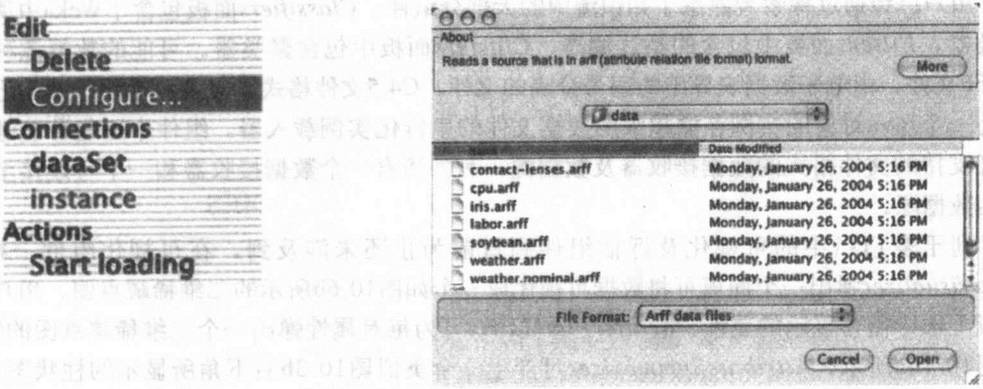


图11-1 知识流界面

现在我们用一个*ClassAssigner*对象来指定哪个属性是类。在*Evaluation*面板上，单击*Evaluation*标签，选定*ClassAssigner*，然后将其置于画布上。下面将数据源与类指定器连接起来，右键单击数据源图标并从菜单中选择*dataset*，如图11-2a所示。一条类似橡皮筋的线会出

现。将鼠标移至类指定器组件并左键单击。一条标记为*dataset*的红线随之出现并将两个组件连接起来。连接好类指定器后，右键单击它来选择类，（从出现的菜单中）选定*Configure*，并键入类属性的位置。



a) 右键单击菜单　　b) 由*Configure*菜单中条目所得到的文件浏览器

图11-2　配置一个数据源

我们用*J48*分类器来进行交叉验证。在数据流模型中，我们首先连接*CrossValidationFoldMaker*，生成一些可供分类器运行的折，然后将其输出结果传递给代表*J48*的一个对象。*CrossValidationFoldMaker*可在*Evaluation*面板上找到。将其选定，置于画布上，再通过右键单击类指定器及从菜单（该菜单与图11-2a中的很相似）中选择*dataset*的方式将*CrossValidationFoldMaker*与类指定器连接起来。下一步在*Classifiers*面板上选定*J48*，再将一个*J48*组件置于画布上。因为有很多不同类型的分类器，用户必须滚动工具条才能找到*J48*。像前面一样将J48与交叉验证折生成器连接起来，但这一次要做双重连接，从交叉验证折生成器弹出的菜单中首先选择*trainingSet*，然后选*testSet*。下一步在*Evaluation*面板上选择*ClassifierPerformanceEvaluator*，从右键单击*J48*弹出菜单中选择*batchClassifier*，将*ClassifierPerformanceEvaluator*与*J48*连接起来。最后，在*Visualization*工具条上把一个*TextViewer*组件置于画布上，性能评估器从弹出的菜单中选择*text*，将分类器性能评估器与*TextViewer*连接起来。

到目前为止，除了还缺少图形查看器外，其他配置与图11-1所示一样。从ARFF载入器的弹出菜单中（图11-2a）选择Start loading，开始流程的运行。对于一个小型数据集，运行过程很快就完成了，如果输入的数据很大，用户就会看到一些动画图标，例如，*J48*上的树开始生长，以及性能评估器上的对勾标记开始闪动。运行过程信息出现在界面底部的状态栏中。从文本查看器的弹出菜单中选择*Show results*，在一个单独的窗口中会看到与探索者中看到的同样形式的交叉验证结果。

欲将本例演示完整，加入一个*GraphViewer*并将它与*J48*图形输出相连接，可以看到为交叉验证的每个折所生成的树的图形化表现形式。加入这个额外的组件并重新进行交叉验证后，428~429
从弹出菜单中选择*Show results*，即可生成一列树，每棵树分别对应交叉验证的一个折。通过生成交叉验证的折并将它们传递给分类器，知识流模型提供了一种方式将所得结果与每个折联系起来。探索者无法做到这一点：探索者只是将交叉验证看作一种应用于分类器的输出结

果的评估方法。

11.2 知识流组件

用户已经通过探索者熟悉了知识流中的大部分组件。*Classifiers*面板包含了Weka中所有的分类器，*Filters*面板中包含的是过滤器，*Clusters*面板中包含聚类器。可能的数据源种类有ARFF文件，由电子数据表导出的逗号分隔的文件，C4.5文件格式，以及一个用于载入已经被作为一个Java对象的实例存储起来的数据文件的串行化实例载入器。组件中还包含能够被探索者支持的文件格式的数据接收器及数据源。另外还有一个数据接收器和一个可连接至数据库的数据源。

列于表11-1中的可视化及评估组件到目前为止还未涉及到。在可视化组件一栏中，*DataVisualizer*弹出一个面板可将数据可视化成一个如图10-6b所示的二维稀疏点图，用户可在面板上选择需要显示的属性。*ScatterPlotMatrix*可为每对属性弹出一个二维稀疏点图的矩阵，如图10-16a所示。*AttributeSummarizer*可产生一个类似图10-3b右下角所显示的柱状图矩阵，其中每一个柱状图对应一个属性。*ModelPerformanceChart*用于绘制ROC曲线及其他阈曲线。*GraphViewer*可弹出一个如图10-6a所示的用于可视化树状模型的面板。如前所述，用户可缩放，摇动及可视化节点上的实例数据（如果已经学习算法存储下来的话）。

表11-1 可视化和评估组件

名称		功能
可视化	DataVisualizer	在一个二维稀疏点阵中可视化数据
	ScatterPlotMatrix	稀疏点阵矩阵
	AttributeSummarizer	一组柱状图，每个柱状图对应一个属性
	ModePerformanceChart	绘制ROC及其他阈曲线
	TextViewer	将数据或模型可视化为文本
	GraphViewer	可视化树状模型
	StripChart	显示一个可滚动的数据点阵
评估	TrainingSetMaker	由数据集生成一个训练集
	TestSetMaker	由数据集生成一个测试集
	CrossValidationFoldMaker	将一个数据集分割成不同的折
	TrainTestSplitMaker	将一个数据集分割成训练和测试集
	ClassAssigner	将属性中的某一个指定为类
	ClassValuePicker	为肯定类选择一个值
	ClassifierPerformanceEvaluator	为批量评估搜集评估统计数据
	IncrementalClassifierEvaluator	为递增评估搜集评估统计数据
	ClustererPerformanceEvaluator	为聚类器搜集评估统计数据
430	PredictionAppender	将分类器的预测附加到一个数据集中

*StripChart*是一个新的专门用于递增学习的可视化组件。如果与下面要讨论的*IncrementalClassifierEvaluator*联合起来使用，*StripChart*可显示一个用于图示精确度的学习曲线，既显示百分比精确度，又可给出均方根概率误差，二者都与时间相对应。它所显示的是一个可水平滚动显出最新结果的固定大小的时间窗口。

*Evaluation*面板所含有的组件列于表11-1的下半部分。*TrainingSetMaker*和*TestSetMaker*可将一个数据集变成一个相应种类的集。*CrossValidationFoldMaker*从一个数据集中构建交叉验证

的折；*TrainTestSplitMaker*通过保留数据集中的部分数据用于测试，将数据集分割成训练集和测试集。*ClassAssigner*使得用户可以选择以哪个属性作为类。用户可用*ClassValuePicker*选择一个值，该值在产生ROC及其他阈曲线时可作为肯定类。*ClassifierPerformanceEvaluator*搜集评估统计数据：它能将文本评估传给文本查看器以及将阈曲线制成性能图表。*IncrementalClassifierEvaluator*有着与递增分类器一样的功能：它可计算运行方差，等等。*ClustererPerformanceEvaluator*与*ClassifierPerformanceEvaluator*很相似。*PredictionAppender*以一个分类器和一个数据集作为输入，并将分类器的预测附加到数据集中。

11.3 配置及连接组件

用户可通过单独配置每个组件，然后将它们连接起来的方式生成知识流。图11-3列出了右键单击不同的组件可得到的典型的操作功能。图中的菜单可分为三部分：*Edit, Connections*和*Actions*。*Edit*操作可删除组件及打开组件的配置面板。配置分类器和过滤器的方式与它们在探索者中的配置方式一样。数据源是通过打开一个文件（与我们前面看到的一样）进行配置的，配置评估组件则需要设定类似交叉验证的折的数量等参数。*Actions*操作则需视具体的组件类型而定，比如从一个数据源开始载入数据，或打开一个窗口显示可视化结果等。
*Connections*操作用于将组件连接到一起，连接的方式是从源组件上选择连接类型，然后再单 431
击目标对象。不是所有的目标都是可连接的：可以被连接的目标是高亮显示的。除非接收连接的目标组件确认所收到的连接是可行的，否则连接菜单中的条目会被禁止（设成灰色）。

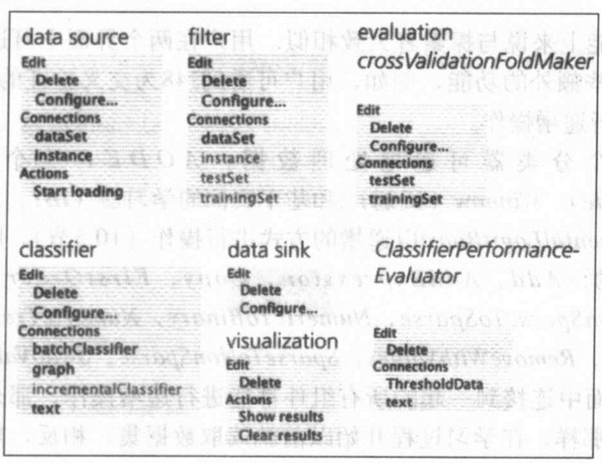


图11-3 知识流组件的操作

有两种可由数据源开始的连接：*dataset*连接和instance连接。前者用于批量操作，适用于像*J48*一样的分类器；后者则用于类似*NaiveBayesUpdateable*的流操作。数据源组件无法同时提供两种类型的连接：一旦一种连接被选定，另外一种就会被禁止。当一个*dataset*连接指向一个批量分类器时，该分类器需要知道该连接指向的是一个训练集还是测试集。要做到这一点，用户必须事先用*Evaluation*面板中的*TestSetMaker*或*TrainingSetMaker*组件将数据源变成一个测试集或训练集。另一方面，一个指向递增分类器的*instance*连接则是直接建立的，因为该流动的实例以递增的方式更新分类器，在训练和测试之间没有分别。在这种情况下，预测是基于每个输入的实例做出，然后并入测试结果的；接下来，该分类器再根据这个实例进行训

练。如果用户将一个*instance*连接指向一个批量分类器，它将会当作一个测试实例，因为训练无法递增进行，而测试则可以。相反，用一个*dataset*连接在批量模式下测试一个递增分类器是完全可行的。

当一个过滤器组件收到来自一个数据源的输入时，它的连接将被激活，*dataset*或*instance*
432 连接即可随之建立。*Instance*连接不可指向有指导过滤器，或那些无法递增处理数据的无指导过滤器（例如*Discretize*）。要从一个过滤器得到一个测试集或训练集，用户必须给该过滤器输入相应类型的数据集。

分类器菜单中有两种类型的连接。第一种是*graph*和*text*连接，可用图形及文本的方式显示分类器的学习状态，且仅仅在该分类器收到一个训练集输入时才被激活。另外一种连接称为*batchClassifier*和*incrementalClassifier*，该种连接为性能评估器提供数据，并且是在输入一个测试集时才可激活。哪种类型的连接被激活取决于该分类器的类型。

评估组件是一个大杂烩。*TrainingSetMaker*和*TestSetMaker*将一个数据集变成一个训练集或测试集。*CrossValidationFoldMaker*把一个数据集分成一个训练集和一个测试集。*Classifier Performance Evaluator*（在第11.1节的范例中使用过）为可视化组件产生文本及图形式输出。其他评估组件操作与过滤器相类似：它们根据输入数据随后分别激活*dataset*，*instance*，*training set*、或*test set*连接（例如，ClassAssigner为一个数据集指定一个类）。

尽管一些可视化组件可做出一些类似Show results及Clear results的动作，它们却不含连接。

11.4 递增学习

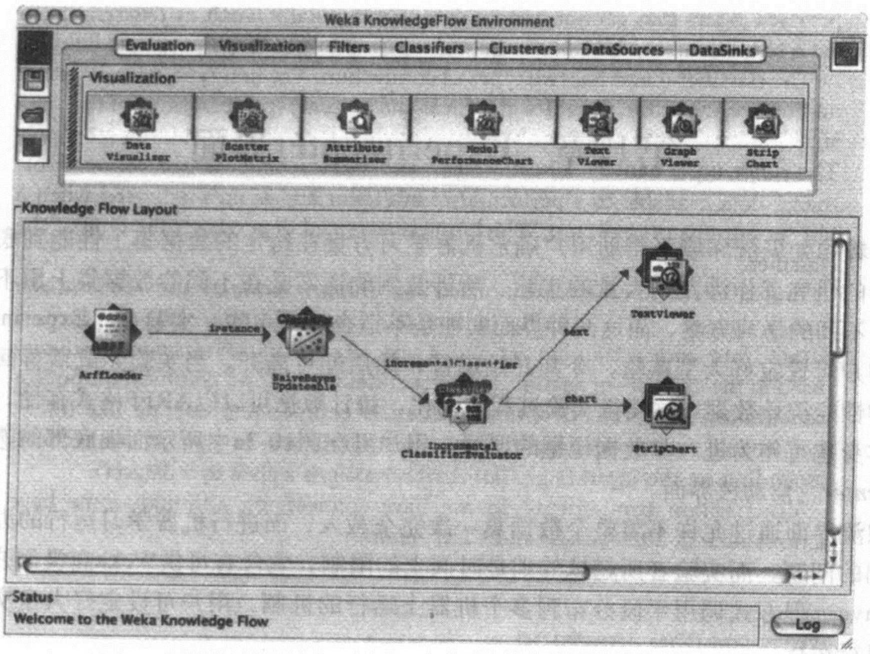
知识流界面从功能上来说与探索者大致相似：用户在两个界面上可进行类似的操作。知识流界面还提供了一些额外的功能，例如，用户可看到J48为交叉验证的每个折所生成的树。但它真正的强项是进行递增操作。

Weka有好几个分类器可递增处理数据：*AODE*，一个朴素贝叶斯版本（*NaiveBayesUpdateable*），*Winnow*（甄别），和基于实例的学习器（*IB1*、*IBk*、*KStar*、*LWL*）。元学习器*RacedIncrementalLogitBoost*以递增的方式进行操作（10.5节）。所有可逐一处理实例的过滤器都是递增的：*Add*，*AddExpression*，*Copy*，*FirstOrder*，*MakeIndicator*，*MergeTwoValues*、*NonSparseToSparse*，*NumericToBinary*，*NumericTransform*，*Obfuscate*，*Remove*，*RemoveType*，*RemoveWithValues*，*SparseToNonSparse*，*SwapValues*。

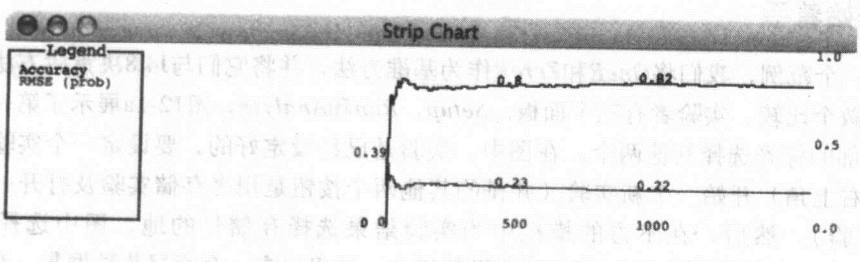
如果在知识流界面中连接到一起的所有组件都能进行递增操作，那么最终的学习系统也可以。它不像探索者那样，在学习过程开始以前就读取数据集。相反，数据源组件对输入实例进行逐个读取并传入知识流链。

图11-4a显示了一个进行递增操作的配置。一个*instance*连接从载入器指向了一个可更新朴
433 素贝叶斯分类器。该分类器的文本输出传给了可显示该模型文本描述的查看器。另外，一个*incrementalClassifier*连接指向了相应的性能评估器。这样就生成了一个*Chart*类型的输出，该输出以管道的方式传给了一个条状图表可视化组件以产生一个可滚动数据点图。

图11-4b给出了条状图表输出。它显示了与时间相对应的精确度和均方根概率误差。随着时间的推移，整个点状图（包括坐标轴）会向左移动以便在右侧为新数据让出空间。当代表时间点0的垂直轴无法再向左移动时，它会停止并且它的起始时间点由0开始增加以便与右侧的数据保持同一步调。因此当该图表呈满屏状态时，它显示的是一个最接近当前时间单位的窗口。该条状图表可被配置从而改变x坐标轴上显示的实例数量。



a) 配置



b) 条状图输出

图11-4 一个递增操作的知识流

这种具体的知识流配置可处理任意大小的输入文件，即使是那些计算机的主存储器无法容纳的文件也可以。然而，这取决于分类器是如何进行内部操作的。例如，即使分类器是递增的，许多基于实例的学习器还是将整个数据集存储在学习器中。 434~435

第12章 Experimenter界面

探索者和知识流环境可帮助用户确定机器学习方案在给定的数据集上性能到底如何。但严肃认真的研究工作涉及到大量的实验，所涉及到的通常是在不同的数据集上用不同的参数设置运行不同的学习方案，而这两种界面实在是不适合这项工作。实验者（Experimenter）界面使得用户可设定好大型实验，令其开始运行，然后即可离开，当实验完成后再回来分析所搜集到的性能统计数据。这使得实验过程自动化。统计数据可以以ARFF格式存储，且这些统计数据本身也可作为进一步数据挖掘的课题。用户可在图10-3a中所示面板底部的选项中选择*Experimenter*，启动该界面。

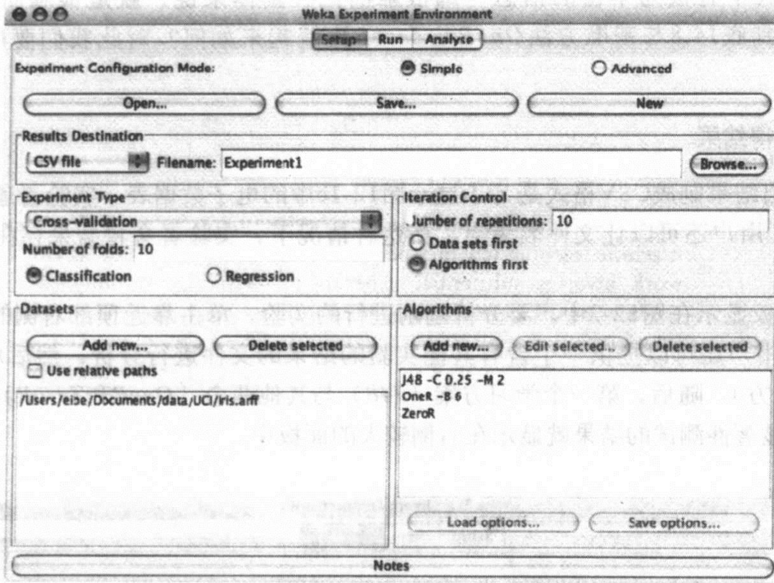
知识流界面通过允许不将整个数据集一次完全载入，而进行机器学习运行的办法来解决有限空间的问题，而实验者所超越的则是时间上的限制。它含有可供Weka高级用户将运算负荷通过Java远程方式调用手段分布到多个机器上运行的机制。用户可设定好大型实验，然后
437 让它们自行运行。

12.1 开始着手

作为一个范例，我们将*OneR*和*ZeroR*作为基准方法，并将它们与J4.8决策树方法通过鸢尾花数据集做个比较。实验者有三个面板：*Setup*，*Run*和*Analyze*。图12-1a展示了第一个：用户可通过顶部的标签选择其他两个。在图中，实验是已经设定好的。要设定一个实验，首先单击*New*（右上角）开始一个新实验（并排的其他两个按钮是用来存储实验及打开一个以前存储过的实验）。然后，在下方的横栏中为实验结果选择存储目的地，图中选择的文件是*Experiment1*，并选择*CSV file*。再往下选择数据集，这里只有一个鸢尾花数据集。在数据集框的右侧，选择将要测试的算法，有三个。单击*Add new*按钮可得到一个标准的Weka对象编辑器，从中用户可选择并配置一个分类器。重复这个操作将图中的三个分类器添加进来。至此，实验已设定好。图12-1a中的其他设置都是默认的。若想重新配置一个已经在列表中的分类器，用户可点击*Edit selected*按钮。用户还可以将用于某个具体分类器的选项以 XML的格式存储起来供以后使用。

12.1.1 运行一个实验

要运行一个实验，单击*Run*标签，弹出的面板上有一个*Start*按钮（及其他一些组件）；单击它。操作进行完毕后会显示一个简单的报告。实验结果存在文件*Experiment1.csv*中。文件的前两行显示在图12-1b中：它们是CSV（逗号分隔的值——译者注）格式，且可直接读入一个电子数据表，所形成的数据表格的前面部分显示在图12-1c中。每一行代表一个10折交叉验证中的1折（见*Fold*列）。交叉验证为每个分类器（见*Scheme*列）运行10次（见*Run*列）。因此该文件中每个分类器对应100行，共300行（还要再加上一行文件的头）。每行都含有大量信息，实际上是46列，包括提供给机器学习方案的选项；训练和测试实例的数量；被正确及错误分类以及未被分类的实例的数量（和百分比）；平均绝对误差，均方根差及很多其他信息。

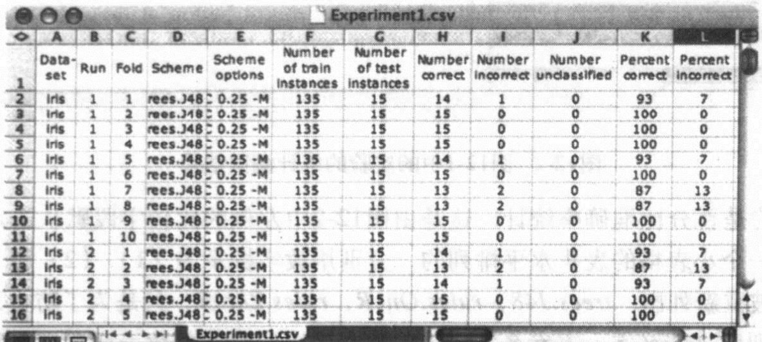


a) 设定实验

```
Dataset,Run,Fold,Scheme,Scheme_options,Scheme_version_ID,Date_time,Number
_of_training_instances,Number_of_testing_instances,Number_correct,Number_
incorrect,Number_unclassified,Percent_correct,Percent_incorrect,Percent_u
nclassified,Kappa_statistic,Mean_absolute_error,Root_mean_squared_error,R
elative_absolute_error,Root_relative_squared_error,SF_prior_entropy,SF_sc
heme_entropy,SF_entropy_gain,SF_mean_prior_entropy,SF_ mean_scheme_entrop
y,SF_mean_entropy_gain,KB_information,KB_mean_information,KB_relative_info
rmation,True_positive_rate,Num_true_positives,False_positive_rate,Num_fal
se_positives,True_negative_rate,Num_true_negatives,False_negative_rate,Nu
m_false_negatives,IR_precision,IR_recall,F_measure,Time_training,Time_tes
ting,Summary,measureTreeSize,measureNumLeaves,measureNumRules

iris,1,1,weka.classifiers.trees.J48,'-C 0.25 -M 2',-217733168393644444,2.
00405230549E7,135.0,15.0,14.0,1.0,0.0,93.33333333333333,6.66666666666666
7,0.0,0.9,0.0450160137965016,0.1693176548766098,10.128603104212857,35.9176
98581356284,23.77443751081735,2.632715099281766,21.14172241153558 2,1.5849
625007211567,0.17551433995211774,1.4094481607690388,21.615653599867994,1.
4410435733245328,1363.79589990507,1.0,5.0,0.0,0.0,1.0,10.0,0.0,0.0,1.0,1.
0,1.0,0.0070,0.0,'Number of leaves: 4\nSize of the tree: 7\n',7.0,4.0,4.0
```

b) 结果文件



	A	B	C	D	E	F	G	H	I	J	K	L
1	Data-set	Run	Fold	Scheme	Scheme options	Number of train instances	Number of test instances	Number correct	Number incorrect	Number unclassified	Percent correct	Percent incorrect
2	iris	1	1	rees.J48	0.25 -M	135	15	14	1	0	93	7
3	iris	1	2	rees.J48	0.25 -M	135	15	15	0	0	100	0
4	iris	1	3	rees.J48	0.25 -M	135	15	15	0	0	100	0
5	iris	1	4	rees.J48	0.25 -M	135	15	15	0	0	100	0
6	iris	1	5	rees.J48	0.25 -M	135	15	14	1	0	93	7
7	iris	1	6	rees.J48	0.25 -M	135	15	15	0	0	100	0
8	iris	1	7	rees.J48	0.25 -M	135	15	13	2	0	87	13
9	iris	1	8	rees.J48	0.25 -M	135	15	13	2	0	87	13
10	iris	1	9	rees.J48	0.25 -M	135	15	15	0	0	100	0
11	iris	1	10	rees.J48	0.25 -M	135	15	15	0	0	100	0
12	iris	2	1	rees.J48	0.25 -M	135	15	14	1	0	93	7
13	iris	2	2	rees.J48	0.25 -M	135	15	13	2	0	87	13
14	iris	2	3	rees.J48	0.25 -M	135	15	14	1	0	93	7
15	iris	2	4	rees.J48	0.25 -M	135	15	15	0	0	100	0
16	iris	2	5	rees.J48	0.25 -M	135	15	15	0	0	100	0

c) 含有结果的电子数据表

图12-1 一个实验

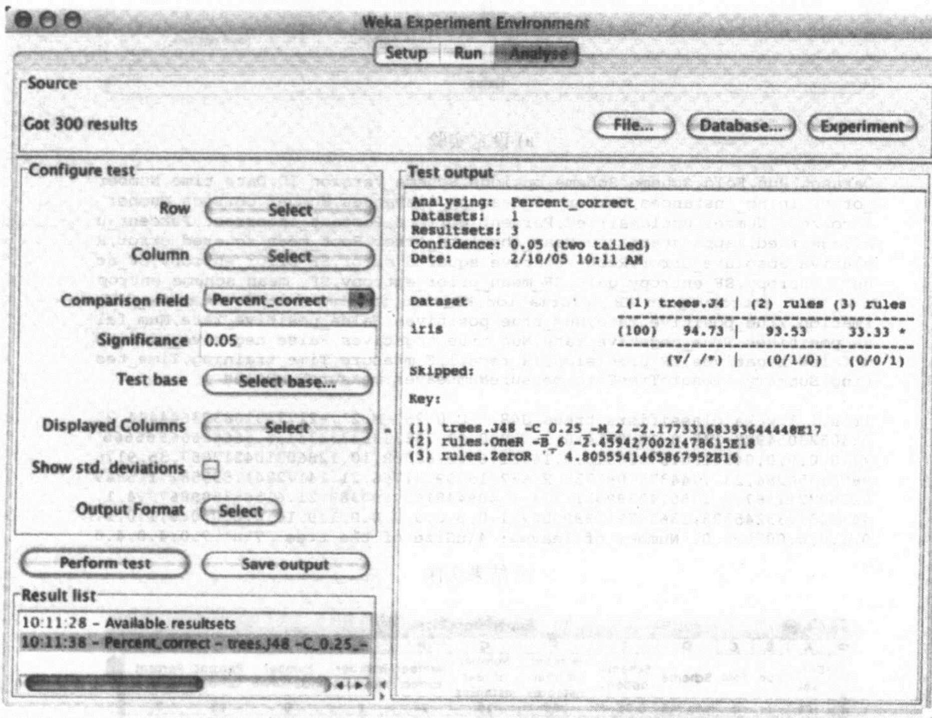
438 ~ 439

电子数据表中有极为丰富的信息，却很难消化。具体来说，就是很难回答上面提到的这个问题：到底J4.8与基准方法*OneR*及*ZeroR*比较起来如何？对此我们要用到*Analyze*面板。

12.1.2 分析所得结果

我们将输出结果制成CSV格式是为了显示图12-1c中的电子数据表。实验者通常将输出制成ARFF格式。用户还可以让文件名为空，在这种情况下，实验者会将结果存在一个临时文件中。

*Analyze*面板显示在图12-2中。要分析刚刚进行的实验，单击靠近顶部右侧的*Experiment*按钮；否则，用户还可以提供一个含有其他实验的结果的文件进行分析。然后单击*Perform test*（靠近左下方）。随后，第一个学习方案（*J48*）与其他两个（*OneR*和*ZeroR*）相比较的有关性能的统计显著性测试的结果就显示在右侧较大的面板中。



440 图12-2 图12-1中的实验的统计测试结果

我们比较的是百分比准确率统计：这是由图12-2中左侧的比较字段默认选定的。三种学习方案是作为一个小表格的表头水平排列的，分别用数字标示为（1）、（2）和（3）。表中列的标签在下方被重新列出：*trees.J48*，*rules.OneR*，*rules.ZeroR*，只是为了防止万一表头中没有足够的空间列出这些名字。方案名字旁边的那些难以预测的整数表明所用方案的版本。默认条件下它们都会列出，以避免在使用同一算法的不同版本所生成的结果之间产生混淆。在鸢尾花（*iris*）行开始的括号中的值（100）是实验的运行次数：10次10折交叉验证。

三个方案的百分比正确率显示在图12-2中：方案1是94.73%，方案2是93.53，和方案3是33.33%。结果旁边的符号（v和*）指明该结果在所指定的统计显著性水平上（当前是0.05，或者5%），好于（v）还是差于（*）基准方案，在本例中基准方案是J4.8。本例中所用的是5.5节中的纠正重复取样t测试。这里，方案3明显要比方案1差，因为方案3的成功率旁边有星形标记。在第2列和第3列的底部是方案运行次数的计数数量（x/y/z），其中，（x）表示该方案在本实验中的数据集上的性能好于基准方案的次数，（y）表示二者性能一样，（z）是其性能差于基准方案的次数。在本例中只用了一个数据集，方案2有1次与方案1（基准方案）相等，而方案3则有1次比方案1差。（注释（v/ /*）放在第一列的底部，帮助用户记忆三种计数数量x/y/z的含义。）

12.2 简单设置

在图12-1a中的*Setup*面板上，我们在大多数选项上都使用了默认值。图中的实验类型是10折交叉验证，重复10次。用户可在左侧中部的框中改变折的数量，及在右侧中部的框中更改重复次数。实验类型是分类；用户也可将其指定为回归。用户还可以选择一个以上的数据集，这样的话，每种算法就会依次应用于每个数据集，并可通过*Data sets first*和*Algorithm first*按钮改变循环运行的顺序。旁置法（holdout）可作为交叉验证的替代方法。它有两种变体，取决于实验时是保留数据集的顺序还是将数据随机化。用户可指定分割百分比（默认是2/3作为训练集，1/3作为测试集）。

实验的设置可以存储起来重新启用。用户可按下*Notes*按钮，弹出一个编辑窗口，对所做的设置进行注释。严谨的Weka用户会很快发现，在打开一个实验后需要对其进行修改，也许是换一个新的数据集或一个新的学习算法。能够避免重新计算以前已经取得过的结果当然要好！况且，如果所得结果能够存储在数据库而不是ARFF或CSV文件中就更好，这正是 441
我们现在要做的。在结果目的地选择器中选择*JDBC database*，连接到任何带有JDBC驱动器的数据库。用户在这里需要给出数据库的URL并填写用户名及密码。要想使这一过程也能用于自己的数据库，用户需要修改Weka分布包中的*weka/experiment/DatabaseUtils.props*文件。如果用户对一个使用数据库的实验做改动，只要以前计算过的结果还能从数据库中取出来，Weka就会沿用它们。这大大简化了通常情况下可看作是数据挖掘研究特征的循环往复的实验过程。

12.3 高级设置

实验者有一种高级模式。单击图12-1a面板顶部（Advanced）则得到如图12-3所示的超强版本的面板。该面板含有更多的控制实验的可选选项，例如，包括生成学习曲线的能力等。然而，高级模式较难运用，且简单模式已经可以满足大多数要求。例如，在高级模式中，用户可设置一个循环用于测试一种算法在一系列不同参数值条件下的性能，但同样的事情也可在简单模式下，通过对该算法进行多次测试，每次使用不同的参数值的方式完成。也许用户真正用得到高级模式的地方是设置一个分布式实验，我们会在第12.5节中进行探讨。

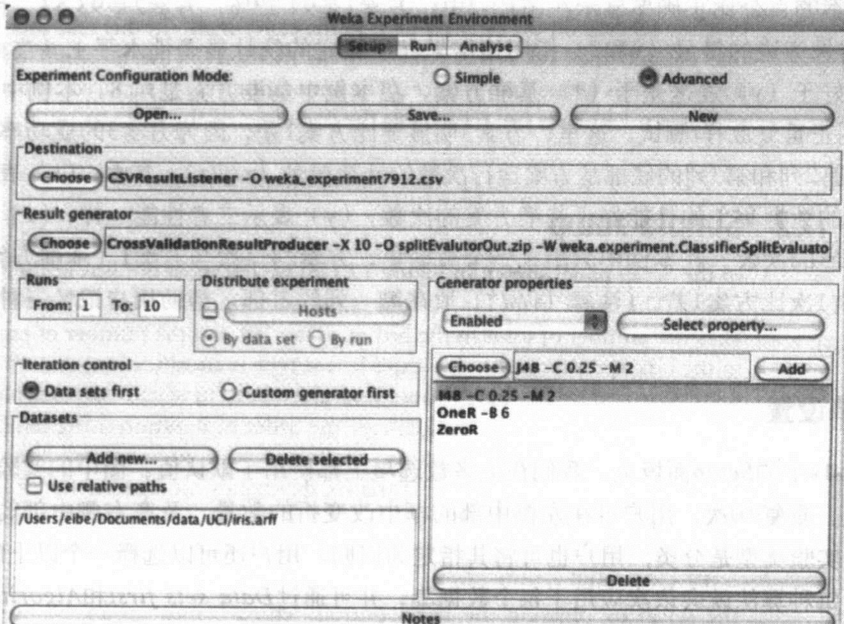


图12-3　在高级模式中设置一个实验

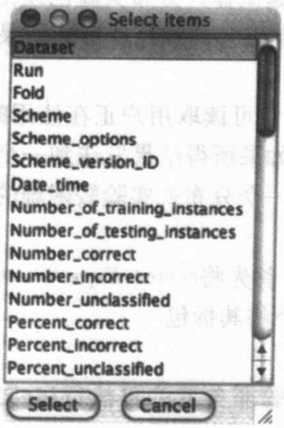
12.4　分析面板

在前面的讨论中，我们使用*Analyze*面板进行了一种学习方案（J48）与其他两种方案
441 （*OneR*和*ZeroR*）相比较的统计显著性测试。该测试基于误差率，即由图12-2中的*Comparison field*决定的。其他类型的统计则可由下拉菜单中选取：百分比错误，百分比未分类，均方根误差，表5-8中的其他误差测量方法，以及不同的熵值。还有，用户可选中*Show std deviations*复选框查看被评估的属性的标准差。

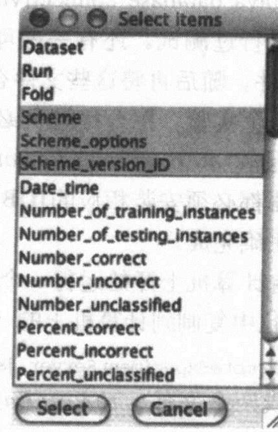
使用*Select base*菜单可将基准方案由J4.8改为其他任何一种学习方案。例如，选中*OneR*意味着将其他方案与*OneR*进行比较。实际上，这只能说明*OneR*与*ZeroR*在统计显著性上有区别，而*OneR*与*J48*之间却没有。除了学习方案以外，*Select base*菜单中还有其他两个选项：*Summary*和*Ranking*。前者将每种学习方案与其他方案逐一进行比较并列出一个矩阵，矩阵的每个格显示的是数据集的数量。在这些数据集上，一个学习方案的性能要明显好于其他方案。后者根据代表赢（>）和输（<）的数据集的总数对学习方案进行排序，并列出一个队列表。队列表的第一列显示的是赢与输的数量之差。

*Row*和*Column*信息组决定了比较矩阵的维数。单击*Select*得到一列已经在实验中测量过的所有属性，即图12-1c中电子数据表的每一列的表头。用户可任意挑选其中的项目作为矩阵的行和列。（用户所做出的选择不会出现在*Select*框中，这是因为可同时选定的参数不止一个。）图12-4列出了
442~443 共有哪些项目被选中作为图12-2中的行和列。图中的两个列表显示了实验所用的参数（即电子数据表中的列）。*Dataset*被选定为行（在本例中，行只有一个，即鸢尾花数据集），而*Scheme*，*Scheme options*和*Scheme_version_ID*被选为列（与通常一样，按住shift键同时单击鼠标可选定多个项目）。全部三个选项都可在图12-2中看到。实际上，它们在底部的相关方案中看得更加清楚。

如果将行和列的选项互换，并再次按下*Perform test*按钮，矩阵将会翻转，所得结果见图12-4c。这里有三行，每行对应一种算法，唯一的一列对应着所用的数据集。如果把行由*Dataset*换成*Run*，再重新进行测试，结果则如图12-4d所示。*Run*指的是交叉验证的运行次数，共有10次，所以有10行。每一行的行标后面的括号中的数字（图12-4c中的100和图12-4d中的10）是对应着该行的结果的数量，即显示在所对应行的每个格中的平均值时，参与计算的所有测量值的数量。还有一个按钮用户可用来同时选择列中的子集加以显示（代表着基准算法的列总是包含在其中），另外一个按钮用来选择输出格式：文本文档（默认），用于LaTeX排版系统的输出格式以及CVS格式。



a) 行选项



b) 列选项

```
Dataset                              (1) iris |
                                   -----------------
trees.J48 '-C 0.25 -M 2'  (100)     94.73 |
rules.OneR '-B 6'         (100)     93.53 |
rules.ZeroR ''            (100)     33.33 |
                                   -----------------
                                    (v/ /*) |
```

c) 互换行与列选项后的结果

```
Dataset      (1) trees.J4 | (2) rules (3) rules
           ----------------------------------------
1            (10)   96    |  94          33.33 *
2            (10)   94    |  94          33.33 *
3            (10)   94    |  94          33.33 *
4            (10)   95.33 |  92.67       33.33 *
5            (10)   95.33 |  94.67       33.33 *
6            (10)   95.33 |  94          33.33 *
7            (10)   94    |  92.67       33.33 *
8            (10)   94    |  92          33.33 *
9            (10)   94    |  93.33       33.33 *
10           (10)   95.33 |  94          33.33 *
           ----------------------------------------
                 (v/ /*)  | (0/10/0)  (0/0/10)
```

d) 将*Dataset*替换成*Run*作为行

图12-4 图12-2中的行和列

12.5 将运行负荷分布到多个机器上

实验者的一个显著优点是它能将一个实验分割并分布到几个处理器上。这只适合高级用户，且只能在*Setup*面板的高级版本中操作。由于从简单版本切换到高级版本时，实验的结构可保持不变，一些用户通过在简单版本设置实验，然后切换到高级版本对其进行分布的方式来避免使用高级版本的面板。无论如何，对一个实验进行分布是高层次的操作，通常是较难的。例如，文件和文件夹权限的设定就比较复杂。

在图12-1a中所示面板上当选择*JDBC database*作为结果目的地，将结果都发送到中央数据库时，分布一个实验可取得最好效果。它使用的是远程手段调用（RMI）机制，且可与任何带有JDBC（Java database connectivity）驱动器的数据库一起工作。分布实验已经在几个免费的数据库中进行过测试。还有一种可行的选择是，用户可令每台机器将各自的结果存储成不同的ARFF文件，随后再将这些文件合并起来。

要分布一个实验，每台终端机必须（1）安装Java，（2）可读取用户正在使用的数据集，（3）正在运行*weka.experiment.RemoteEngine*实验服务器。如果所得结果发送到一个中央数据库，每台机器都必须安装相应的JDBC数据库驱动器。运行一个分布式实验较困难的部分就是将以上这些正确完成。

要在一台计算机上开始运行一个远程引擎实验服务器，首先将*remoteExperimentServer.jar*
444~445 从Weka分布包中复制到计算机上的一个目录中。用以下命令将其拆包

```
jar xvf remoteExperimentServer.jar
```

它扩展成两个文件：*remoteEngine.jar*，一个含有实验服务器的可执行*jar*文件，以及*remote.policy*。

*Remote.policy*文件给远程引擎授予权限令其进行一些特定操作，如连接到一些端口，或存取一个文件夹等。该文件需经过编辑以便在一些需要授予的权限中指明正确的路径；当用户查看该文件时就会不言自明。在默认条件下，它规定所需代码可从Web的任意地址由HTTP 80端口下载，但远程引擎也可从文件URL载入代码。要做到这一点，将该文件中对应文件URL所举的示例解除注释状态并相应地替换路径名。远程引擎也需要能够存取实验中所用的数据集（见*remote.policy*中的第一个条目）。读取数据集的路径是在实验者（即客户端）中指定的，且同样的路径必须也适用于远程引擎的运行条件。要达到这个条件，也许有必要在实验者的*Setup*面板上通过选中*Use relative paths*复选框的方式指定相对路径名。

要启动远程引擎服务器，在含有*remoteEngine.jar*的目录下键入

```
java -classpath remoteEngine.jar:<path_to_any_jdbc_drivers>
     -Djava.security.policy=remote.policy weka.experiment.RemoteEngine
```

如果一切正常进行，用户会看到以下信息（或类似信息）：

```
Host name : ml.cs.waikato.ac.nz
RemoteEngine exception: Connection refused to host:
ml.cs.waikato.ac.nz; nested exception is:
     java.net.ConnectException: Connection refused
Attempting to start rmi registry...
RemoteEngine bound in RMI registry
```

乍看似乎出了问题，实际却是好消息！连接被拒绝是因为该服务器上RMI注册未运行，因此远程引擎启动了一个RMI注册。在所有终端机上重复该过程。在同一台终端机上运行一

个以上的远程引擎是没有意义的。

键入以下命令启动实验者:

```
java -Djava.rmi.server.codebase=<URL_for_weka_code>
     weka.gui.experiment.Experimenter
```

其中的URL指明远程引擎可到哪里找到待执行的代码。如果该URL给出一个目录（即含有Weka目录的目录），而不是一个jar文件，它一定是以路径分隔符（例如，/）结尾。

图12-3中实验者高级*Setup*面板的左侧中部有一个小窗格，它决定了一个实验是否会被分布。该窗格通常是不活动的。要分布该实验，单击窗格中的复选框（*Hosts*按钮左侧——译者 446
注）以激活Hosts按钮；（点击Hosts按钮——译者注）一个窗口会弹出来询问该实验将要分布的终端机器。终端机器的名字必须完整地给出（例如，*ml.cs.waikato.ac.nz*）。

进入终端机后，按通常方式配置实验的其余选项（像前面提到的那样，最好配置完后再切换到高级设置方案中）。使用*Run*面板开始进行实验时，不同终端机器上的子实验的进程也会显示出来，同时显示的还有任何可能的出错信息。

分布一个实验涉及到将其分割成子实验，并通过远程方式调用发送到终端机上执行。默认条件下，实验是按照数据集划分的，即终端机的数量不可能多于数据集的数量。因此每个子实验都是独立的，它将所有的学习方案应用于一个单独的数据集。一个仅含有几个少量数据集的实验也可以按照运行次数划分。比方说，一个10次10折交叉验证可被分割成10个子实验，每次运行一个子实验。 447

第13章 命令行界面

隐藏在Weka的互动式界面探索者、知识流和实验者后面的是Weka的基本功能。这些功能可以原始形式通过命令行界面运行。从图10-3a底部的界面选项中选择*Simple CLI*，得到一个普通文件面板，用户可在面板底部的栏中键入指令。另外，用户还可在操作系统的命令行界面上直接运行*weka.jar*文件中的类，要想以这种方式运行，用户必须按照Weka的*README*文件中说明的那样，首先设定*CLASSPATH*环境变量。

13.1 开始着手

在10.1节的开始，我们利用探索者在天气数据上运行J4.8学习器。要在命令行界面做同样的事情，在文件面板底部的栏中键入：

449
```
java weka.classifiers.trees.J48 -t data/weather.arff
```

这道咒语调用Java虚拟机（在Simple CLI中，Java已经载入）且指示它运行J4.8。Weka是以包的形式组织起来的，对应着一个目录层级结构。所运行的程序叫做J4.8，它是在*trees*包中，是*Classifiers*的子包，而*Classifiers*则是整个*weka*包的一部分。下一节会介绍更多有关包结构的详细情况。－*t*选项表明随后的参数是训练文件的名字，假设天气数据就存在用户启动Weka的目录下的*data*子目录中。由运行结果合成的文本显示在图10-5中。在Simple CLI界面中，该结果会出现在用户键入命令的栏上方的面板中。

13.2 Weka的结构

我们已经解释过如何在探索者中启用过滤器和学习方案，以及在知识流界面中将它们连接到一起。要想再进一步深入了解，有必要学习Weka是如何构成的。有关Weka的详细且及时得到更新的信息可在分布包的在线文档中找到。该文档要比探索者及知识流的对象编辑器中的*More*按钮所给出的对学习和过滤方案的描述技术程度要高。该文件是通过Sun的Javadoc应用程序有源代码中的注释文本直接产生的。要理解它的结构，用户需要知道Java程序是如何构成的。

13.2.1 类，实例和包

每个Java程序都是作为一个类实现的。在面向对象的编程中，一个类就是变量加上一些在这些变量上进行操作的方法的一个集合。合在一起，它们定义了属于该类的一个对象的行为。一个对象简单说就是一个为其所有的变量赋值的类的实例体。在Java中，一个对象也可称为类的一个实例。不幸的是，这与本书所用的术语相冲突。本书中的class和instance出现在机器学习的不同的上下文中具有截然不同的含义。从现在开始，用户将不得不根据它们的上下文来推测这些术语所真正代表的意思。这并不很难，且有时我们会用*object*这个词来代替Java中的*instance*以避免歧义。

在Weka中，一个具体学习算法的实现是封装在一个类中的。例如，上面描述过的*J48*类会

创建一个C4.5决策树。每次Java虚拟机执行*J48*，它会为创建及存储一个决策树分类器分配存储器，从而生成一个该类的实例。所用的算法，它所创建的分类器，以及用于输出该分类器的程序都是类*J48*的实例体的组成部分。 450

较大的程序通常被分割为一个以上的类。例如，类*J48*就不含任何用于构建决策树的代码。它只含有承担大部分工作的其他类的实例的参考。当类的数量很多时，就像在Weka中一样，会难于理解及查找。Java的包结构可令许多类组织在一起。一个包就是一个含有相关联类的合集的一个目录，例如，上面提到过的包*trees*就含有实现了决策树的类。所有的包根据一个层级结构组织在一起，而该层级结构与目录的层级结构相对应，即，*trees*是*classifiers*包的一个子包，而*classifiers*包本身又是整个Weka包的一个子包。

当使用Web浏览器查看由Javadoc产生的在线文档时，用户首先看到的是按英文字母顺序排列的Weka中所有的包的一个列表（图13-1a）。我们在这里依照重要性先后次序从中挑选几个加以介绍。

13.2.2 weka.core 包

*Core*包是Weka系统的核心，并且它里面的类可被几乎其他所有的类读取。用户可单击*weka.core*超链接，得到图13-1b所示的弹出Web页，看看它们到底是什么。

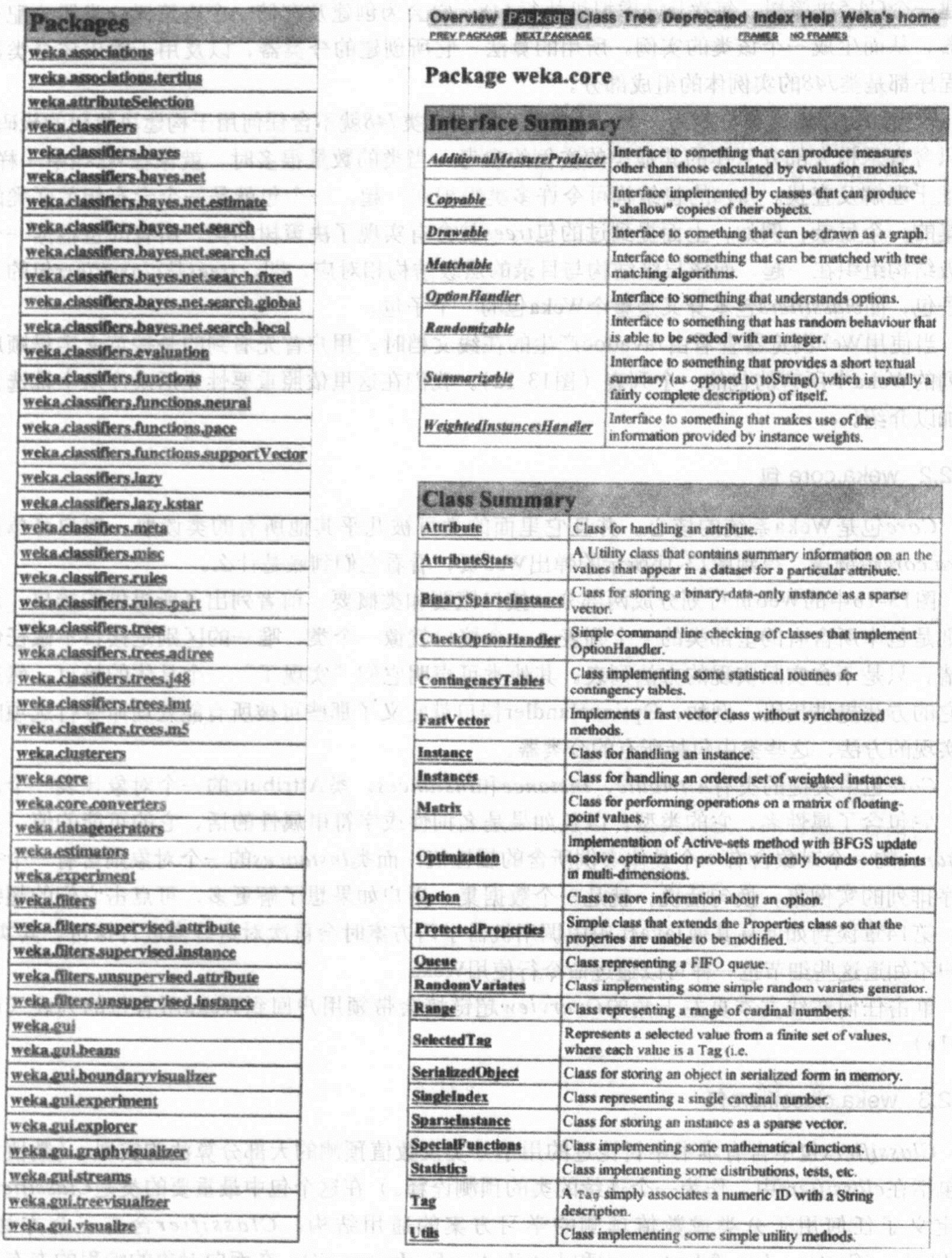
图13-1b中的Web页可划分成两部分：接口概要和类概要。前者列出了所提供的接口，后者则是包中所含有的全部类的一个列表。一个接口就像一个类，唯一的区别是接口不做任何事情，只是不含实际实现的方法列表。其他类可声明它们"实现了"一个具体的接口，然后为它的方法提供代码。例如，OptionHandler接口就定义了那些可被所有能处理命令行选项的类实现的方法，这些类中包括所有的分类器。

*Core*包中关键的类有*Attribute*，*Instance*和*Instances*。类Attribute的一个对象代表一个属性。它包含了属性名，它的类型，以及如果是名词性或字符串属性的话，它的可能的值。类*Instance*的一个对象含有一个具体实例所含的属性值；而类*Instances*的一个对象则含有一个按顺序排列的实例集，换句话说，就是一个数据集。用户如果想了解更多，可点击它们的超链接；第14章谈到如何从其他Java代码中调用机器学习方案时会再次对这些类进行探讨。其实，用户不知道这些细节也一样可以通过命令行使用Weka。

单击任何在线文本页左上角的*Overview*超链接会带领用户回到Weka所有包的列表（图13-1a）。 451

13.2.3 weka.classifiers包

*Classifiers*包中含有本书中讨论过的用于分类及数值预测的大部分算法的实现。（数值预测包括在*classifiers*中，作为一个连续的类的预测诠释。）在这个包中最重要的类是*Classifier*，它定义了任何用于分类或数值预测的学习方案的通用结构。*Classifier*含有三个方法：*buildClassifier()*、*classifyInstance()*和*distributionForInstance()*。在面向对象的编程的专有名词中，学习算法用*classifier*的子类代表，因此，自动继承这三个方法。每种方案都会根据构建分类器以及它对实例进行分类的具体方式对这三个方法进行重新定义。为在其他Java代码中构建及使用分类器提供了一个统一的接口。因此举例来说，同样的评估模块可用来评估Weka中任何分类器的性能。

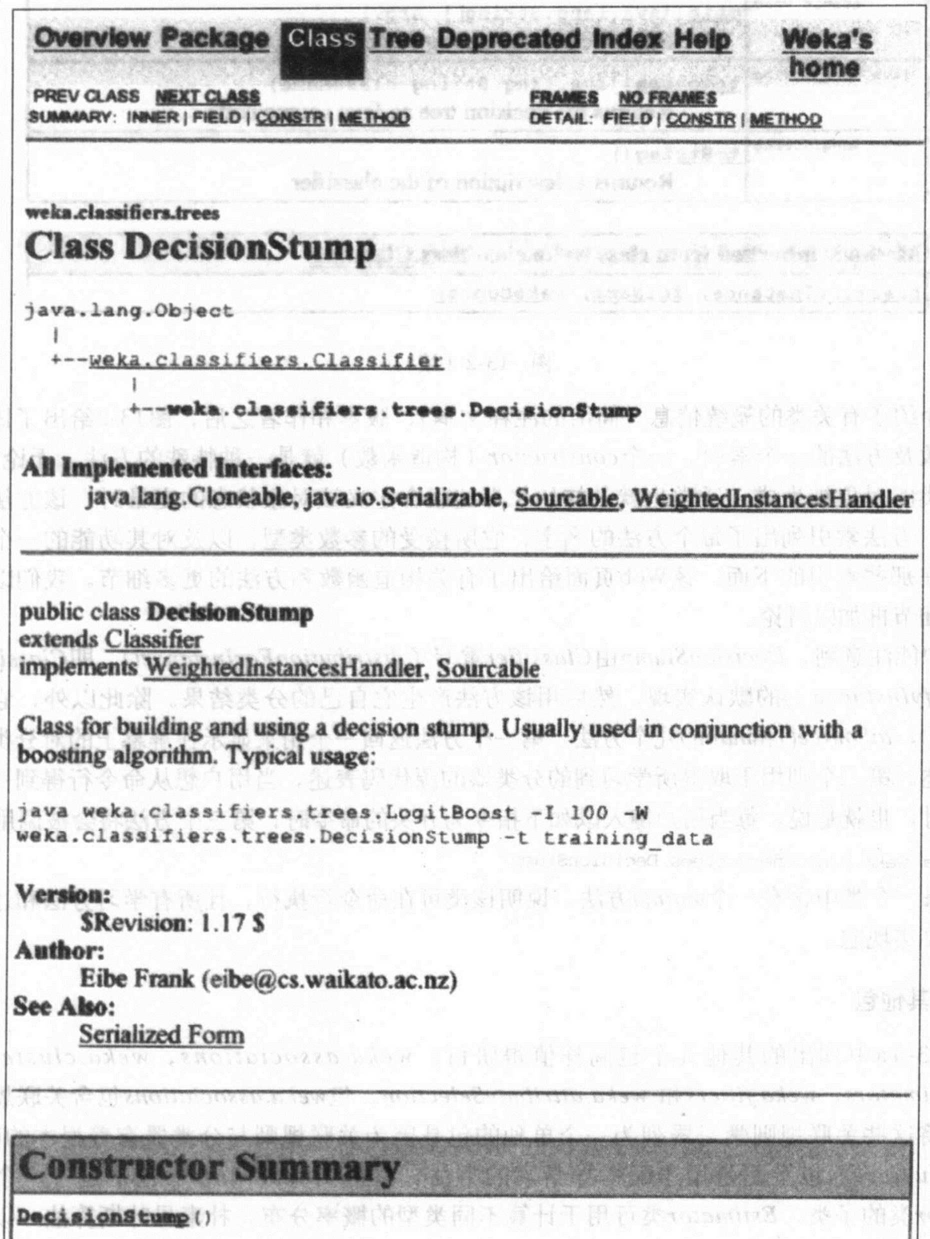


a) 首页　　　　b) *weka.core*包

图13-1　使用Javadoc

如果想看一个例子，单击*weka.classifiers.trees*，然后再单击*DecisionStump*。*DecisionStump*是一个用于构建简单单级二叉决策树（用一个额外的分支代表残缺值）的类。

它的文本页显示在图13-2中，在靠近文本顶部的地方给出了该类的完整有效的名字*weka.classifiers.trees.DecisionStump*。任何时候用户想从命令行构建一个决策树桩，都必须使用这个长长的名字。类的名字会出现在一个小的树结构中，与其同时显示的还有这个类的层级结构中其他相关联的部分。从该结构中用户可观察到，*DecisionStump*是*weka.classifiers.Classifier*的一个子类，而*weka.classifiers.Classifier*本身是*java.lang.Object*的一个子类。在Java中，类Object是最通用的类，所有的类都自动成为它的子类。



Overview Package Class Tree Deprecated Index Help Weka's home

PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

weka.classifiers.trees
Class DecisionStump

```
java.lang.Object
  |
  +--weka.classifiers.Classifier
        |
        +--weka.classifiers.trees.DecisionStump
```

All Implemented Interfaces:
java.lang.Cloneable, java.io.Serializable, Sourcable, WeightedInstancesHandler

public class **DecisionStump**
extends Classifier
implements WeightedInstancesHandler, Sourcable

Class for building and using a decision stump. Usually used in conjunction with a boosting algorithm. Typical usage:

```
java weka.classifiers.trees.LogitBoost -I 100 -W
weka.classifiers.trees.DecisionStump -t training_data
```

Version:
$Revision: 1.17 $
Author:
Eibe Frank (eibe@cs.waikato.ac.nz)
See Also:
Serialized Form

Constructor Summary

DecisionStump()

图13-2 *DecisionStump*: *weka.classifiers.trees*包中的一个类

Method Summary	
void	**buildClassifier**(Instances instances) Generates the classifier.
double[]	**distributionForInstance**(Instance instance) Calculates the class membership probabilities for the given test instance.
static void	**main**(java.lang.String[] argv) Main method for testing this class.
java.lang.String	**toSource**(java.lang.String className) Returns the decision tree as Java source code.
java.lang.String	**toString**() Returns a description of the classifier.

Methods inherited from class weka.classifiers.Classifier
classifyInstance, forName, makeCopies

图　13-2（续）

在介绍了有关类的笼统信息（简洁的注释文本）、版本和作者之后，图13-2给出了该类的构造函数及方法的一个索引。一个*constructor*（构造函数）就是一种特殊的方法，无论何时，当一个类的对象被生成，通常也就是初始化那些联合定义该对象状态的变量时，该方法总会被调用。方法索引列出了每个方法的名字、它所接受的参数类型，以及对其功能的一个简短描述。在那些索引的下面，该Web页面给出了有关构造函数和方法的更多细节。我们以后会对这些细节再加以讨论。

用户能注意到，*DecisionStump*由*Classifier*重写了*distributionForInstance()*，即*Classifier*类中*classifyInstance ()*的默认实现，然后用该方法产生它自己的分类结果。除此以外，它还有*toString()*，*toSource()*和*main()*几个方法。第一个方法返回一个用来显示在屏幕上的对分类器的文本描述。第二个则用于取得所学习到的分类器的源代码表述。当用户想从命令行得到一个决策树桩时，也就是说，每当用户键入以如下指令为开头的命令时，第三个方法将会被调用

```
java weka.classifiers.trees.DecisionStump
```

如果一个类中含有一个*main()*方法，说明该类可在命令行执行，且所有学习方法和过滤器算法都要实现它。

13.2.4　其他包

图13-1a中列出的其他几个包同样值得研讨：*weka.associations*，*weka.clusterers*，*weka.estimators*，*weka.filters*和 *weka.attributeSelection*。包*weka.associations*包含关联规则学习器。将这些关联规则学习器列为一个单独的包是因为关联规则与分类器有着根本的区别。*Weka.clusterers*包里面是用于无指导学习的方法。在*weka.estimators*包中包含一个通用*Estimator*类的子类，*Estimator*类可用于计算不同类型的概率分布。朴素贝叶斯算法（以及其他算法）会用到这些子类。

在*weka.filters*包中，类*Filter*定义了含有过滤器算法的类的一般结构，这些算法都作为

*Filter*的子类实现。与分类器一样，过滤器也可在命令行中使用，我们很快会谈到如何使用。 452~455
*Weka.attributeSelection*包中含有几个用于属性选择的类。*Weka.filters.supervised.attribute*包中的*AttributeSelectionFilter*会用到这几个类，但它们也可被单独调用。

13.2.5 Javadoc索引

正如上面提到的，所有的类都自动成为*Object*的子类。要查看与Weka中类的层级结构相对应的树，从任何在线文本页的顶部选择*Overview*链接。单击*tree*则有一个树的全貌呈现出来，该树显示的是某个类的子类或超类，例如，那些继承了*Classifier*的类等。

在线文档包含了Weka中所有公开使用的变量（称作字段）及方法的一个索引。换句话说，即所有可从用户自己的Java代码中读取的字段和方法。要查看这个索引，单击*Overview*，然后单击*Index*。

假如设用户想知道哪些Weka的分类器和过滤器可进行递增操作，在索引中搜索*incremental*这个字，会很快将用户引导至关键字*UpdateableClassifier*。实际上，这是一个Java接口：接口在展示出全貌的树中被列于类的后面。用户想要找的是实现了该接口的所有类。在任何出现该接口的字样上单击，会得到一个描述该接口并列出了实现该接口的所有分类器的页面。如果用户不知道关键字*StreamableFilter*，要找到想要的过滤器则有点儿难。*StreamableFilter*是一个用于令数据呈“流”式通过过滤器的接口的名字，该页面同样列出了实现该接口的过滤器。如果已经知道了任何可进行递增操作的过滤器的例子，用户会在不经意间发现这个关键词。

13.3 命令行选项

在上面所举的例子中，-t选项用于在命令行中将训练文件的名字与学习算法沟通起来。许多其他选项可用于任何学习方案，也有一些与具体方案关联的选项只能用于某些方案。如果用户启动一个不加任何命令行选项的方案，它会显示所有可用的选项：首先是通用性选项，

```
java weka.classifiers.trees.J48
```

接着是与该方案相关联的选项。在命令行界面中，键入：
用户会看到一列适用于所有学习方案的通用选项，如表13-1所示，以及表13-2中的只适用于*J48*的选项。我们会对通用选项加以说明，然后简单回顾与具体方案相关联的选项。

13.3.1 通用选项

表13-1中的选项确定了哪些数据用于训练，哪些用于测试，分类器是如何评估的，以及显示哪种统计数据。例如，当用一个独立测试集评估学习方案时，-T选项用来提供测试文件的名字。默认条件下，在一个ARFF文件中最后一个属性是类，但用户可用-c后面接所选定的属 456
性位置，将其他属性声明为类，1表示第一个属性，2表示第二个，等等。当进行交叉验证时（如果不提供测试文件，交叉验证是默认选项），数据要首先进行随机混合。如果要重复多次交叉验证，且每次以不同的方式随机混合数据，用-s设定随机数种子（默认是1）。在测试大型数据集时，用户可用-x选项把用于交叉验证的折的数量从默认的10降低至想要的数量。

在探索者中，成本敏感评估可以像第10.1节中描述的那样被调用。要想在命令行中取得同样效果，用-m选项来提供一个含有成本矩阵的文件的名字。以下是一个天气数据的成本矩阵：

```
2   2  % Number of rows and columns in the matrix
0  10  % If true class yes and prediction no, penalty is 10
1   0  % If true class no and prediction yes, penalty is 1
```

表13-1　Weka中用于学习方案的通用选项

选　项	功　能
-t <训练文件>	指定训练文件
-T <测试文件>	指定测试文件。如果该选项为空，就在训练数据上进行交叉验证
-c <类索引>	指定类属性的索引
-s <随机数种子>	指定用于交叉验证的随机数种子
-x <折的数量>	指定用于交叉验证的折的数量
-m<成本矩阵文件>	指定含有成本矩阵的文件
-d <输出文件>	指定模型的输出文件
-l <输入文件>	指定模型的输入文件
-o	只输出统计数据，不输出分类器
-i	对含有两个类的问题输出信息检索统计数据
-k	输出信息理论统计数据
-p <属性区间>	输出测试实例的预测
-v	不对训练数据输出统计数据
-r	输出累积边差分布
-z <类名字>	输出分类器的源表述
-g	输出分类器的图形表述

第一行给出了行和列的数量，也就是类值的数量。接着是损失（penalty）矩阵。由%引导的注释可附加在任何一行的末尾。

457 将模型保存起来以及重新载入是可以做到的。如果用户用-d提供一个输出文件的名字，Weka就会把由训练数据产生的分类器保存下来。要在一个新的批量测试数据上评估同样的分类器，用户可用-l将该分类器重新载入，而无需重新构建。如果该分类器可递增更新，用户还可提供训练文件以及输入文件，Weka会载入该分类器并用所给的训练实例对其进行更新。

如果用户只想检验一个学习方案的性能，用-o选项可禁止模型的输出。用-i可查看精确度，反馈度及F测量（第5.7节）的性能指标，用-k可计算由一个学习方案导出的概率的信息理论指标（第5.6节）。

通常weka用户想要知道学习方案将每个测试实例预测为哪些类值。-p选项会显示出每个测试实例的编号、类、该方案所作预测的置信度，以及所预测的类值。它还为每个实例输出属性值，属性值后面必须跟着它的值区间的详细说明（例如，1－2），如果用户不想要任何属性值，就用0。用户还可以输出训练数据的累积边差分布，该分布显示的是边差指标的分布（7.5节）是如何随着提升迭代的轮次变化的。最后，用户可输出分类器的源表述，并且如果分类器可产生一个图形表述的话，也同样可以显示出来。

13.3.2　与具体方案相关的选项

表13-2列出了只适用于J4.8的选项。用户可强迫算法使用未修剪的树，而不使用修剪过的树。尽管子树上升能够增加效率，用户能够对其加以阻止。用户可设定用于修剪的置信度阈，以及任何叶上的可允许的实例数量的最小值，两个参数在第6.1节中都讨论过。除了可实行C4.5的标准修剪程序外，也可进行减少误差修剪（第6.2节）。-N选项掌管着旁置集的大小，

数据集被平均划分成与该选项的值相等数量的部分，并将最末尾的部分旁置（该选项默认值是3）。用户可用拉普拉斯技巧来平滑概率估计，在选择修剪集时，为数据的随机混合设定种子值，以及存储实例的信息以备将来用作可视化。最后，用-B可为名词性属性构建一个二叉树，而不是通常的多分支树。

表13-2　与J4.8决策树学习器相关的选项

选　项	功　能	
-U	使用未修剪的树	
-C<修剪置信度>	为修剪指定置信度阈	
-M<实例数量>	指定单个叶上实例数量的最小值	
-R	使用减少误差修剪	
-N<折的数量>	指定用于减少误差修剪的折的数量。其中一个折用作修剪集	
-B	只使用二叉分割	
-S	不进行子树上升	
-L	保留实例信息	
-A	用拉普拉斯平滑法来平滑概率估计	458~459
-Q	用于混合数据的种子值	

第14章 嵌入式机器学习

当从图形用户界面或命令行调用学习方案时，用户不需要了解任何有关Java编程的知识。在本节中，我们会讲解如何从用户自己的代码中使用这些算法。使用面向对象编程语言的优势在以下的讲解中会变得更清楚。从现在开始，我们假定用户至少知道一些有关Java的基本常识。在绝大多数数据挖掘的实用程序中，学习组件是一个更大型软件环境中的一个完整部分。如果该软件环境是用Java写成的，用户无需自己编写任何机器学习的代码即可用Weka解决有关的学习问题。

14.1 一个简单的数据挖掘程序

我们会展示一个简单的用于学习模型的数据挖掘程序，该模型将文本文件按两个类别划分：hit和miss。该程序可应用于各种各样的文本：我们将这些文本称为消息。它的实现方法是用第
461 10.3节中提到过的*StringToWordVector*过滤器以在第7.3节中描述过的方式，将消息转换成属性向量。我们假设每次处理一个新文件时该程序都会被调用。如果用户为该文件提供一个类标签，系统就用它作训练；否则，就对其进行分类。决策树分类器*J48*会用来做这项工作。

14.2 讲解代码

图14-1给出了在一个名字叫*MessageClassifier*的类中实现的应用程序的源代码。*main()*方法所接受的命令行可变参数是：一个文本文件的名字（由选项-m给出），持有类*MessageClassifier*的一个对象的文件的名字（-t），和文件中消息的分类（-c），最后一个选项可缺省。如果用户提供一个（消息的）分类，消息会被转换成用于训练的一个例子；如果不提供，则*MessageClassifier*对象会被用于将其分类成hit或miss。

14.2.1 main ()

*main()*方法将消息读入一个Java *StringBuffer*，并查看是否用户为其提供了一个分类。然后它根据-t选项给出的文件读取一个*MessageClassifier*对象。如果该文件不存在，它会生成一个新的*MessageClassifier*类的对象。不论该文件是读入的还是新生成的，它都命名为*messageCl*。如果已经提供了分类，在经过查验确信它不含非法命令行选项后，该程序会调用方法*updateData()*来更新存储在*messageCl*文件中的训练数据；否则就调用*classifyMessage()*对其进行分类。最后，因为*messageCl*对象也许已经发生了变化，它会被重新保存到文件中。在以下的描述中，我们会首先讨论一个新的*MessageClassifier*对象是如何通过构造函数*MessageClassifier()*生成的，然后解释两个方法*updateData()*和*classifyMessage()*是如何工作的。

14.2.2 MessageClassifier()

每当一个新的*MessageClassifier*被生成，用于持有过滤器和分类器的对象就被自动产生。该过程中唯一重要的部分就是生成一个数据集，它是由构造函数*MessageClassifier()*负责完成

的。首先，数据集的名字要作为一个字符串存储起来。然后为每个属性生成一个*Attribute*对象，一个用来持有与文本消息相对应的字符串，另一个则用于持有数据集的类属性。这些对象是存储在一个*FastVector*类型的动态数组中。(*FastVector*是标准Java *Vector* class的Weka自我实现，因为历史原因，该类在Weka中广泛使用。) 462

```
/**
 * Java program for classifying text messages into two classes.
 */

import weka.core.Attribute;
import weka.core.Instance;
import weka.core.Instances;
import weka.core.FastVector;
import weka.core.Utils;
import weka.classifiers.Classifier;
import weka.classifiers.trees.J48;
import weka.filters.Filter;
import weka.filters.unsupervised.attribute.StringToWordVector;
import java.io.*;

public class MessageClassifier implements Serializable {

  /* The training data gathered so far. */
  private Instances m_Data = null;

  /* The filter used to generate the word counts. */
  private StringToWordVector m_Filter = new StringToWordVector();

  /* The actual classifier. */
  private Classifier m_Classifier = new J48();

  /* Whether the model is up to date. */
  private boolean m_UpToDate;

  /**
   * Constructs empty training dataset.
   */

  public MessageClassifier() throws Exception {

    String nameOfDataset = "MessageClassificationProblem";

    // Create vector of attributes.
    FastVector attributes = new FastVector(2);

    // Add attribute for holding messages.
    attributes.addElement(new Attribute("Message", (FastVector)null));
    // Add class attribute.
    FastVector classValues = new FastVector(2);
    classValues.addElement("miss");
    classValues.addElement("hit");
    attributes.addElement(new Attribute("Class", classValues));
```

图14-1 消息分类器的源代码 463

```
    // Create dataset with initial capacity of 100, and set index of class.
    m_Data = new Instances(nameOfDataset, attributes, 100);
    m_Data.setClassIndex(m_Data.numAttributes() - 1);
  }

  /**
   * Updates data using the given training message.
   */
  public void updateData(String message, String classValue) throws Exception {

    // Make message into instance.
    Instance instance = makeInstance(message, m_Data);

    // Set class value for instance.
    instance.setClassValue(classValue);

    // Add instance to training data.
    m_Data.add(instance);
    m_UpToDate = false;
  }

  /**
   * Classifies a given message.
   */
  public void classifyMessage(String message) throws Exception {

    // Check whether classifier has been built.
    if (m_Data.numInstances() == 0) {
      throw new Exception("No classifier available.");
    }

    // Check whether classifier and filter are up to date.
    if (!m_UpToDate) {

    // Initialize filter and tell it about the input format.
    m_Filter.setInputFormat(m_Data);

    // Generate word counts from the training data.
    Instances filteredData  = Filter.useFilter(m_Data, m_Filter);

    // Rebuild classifier.
    m_Classifier.buildClassifier(filteredData);
    m_UpToDate = true;
  }

  // Make separate little test set so that message
  // does not get added to string attribute in m_Data.
  Instances testset = m_Data.stringFreeStructure();

  // Make message into test instance.
  Instance instance = makeInstance(message, testset);

  // Filter instance.
  m_Filter.input(instance);
  Instance filteredInstance = m_Filter.output();
```

464
≀
465

图 14-1（续）

```
    // Get index of predicted class value.
    double predicted = m_Classifier.classifyInstance(filteredInstance);

    // Output class value.
    System.err.println("Message classified as : " +
                       m_Data.classAttribute().value((int)predicted));
  }

  /**
   * Method that converts a text message into an instance.
   */
  private Instance makeInstance(String text, Instances data) {

    // Create instance of length two.
    Instance instance = new Instance(2);

    // Set value for message attribute
    Attribute messageAtt = data.attribute("Message");
    instance.setValue(messageAtt, messageAtt.addStringValue(text));
    // Give instance access to attribute information from the dataset.
    instance.setDataset(data);
    return instance;
  }

  /**
   * Main method.
   */
  public static void main(String[] options) {

    try {

      // Read message file into string.
      String messageName = Utils.getOption('m', options);
      if (messageName.length() == 0) {
        throw new Exception("Must provide name of message file.");
      }
      FileReader m = new FileReader(messageName);
      StringBuffer message = new StringBuffer(); int l;
      while ((l = m.read()) != -1) {
        message.append((char)l);
      }
      m.close();

      // Check if class value is given.
      String classValue = Utils.getOption('c', options);

      // If model file exists, read it, otherwise create new one.
      String modelName = Utils.getOption('t', options);
      if (modelName.length() == 0) {
        throw new Exception("Must provide name of model file.");
      }
      MessageClassifier messageCl;
      try {
```

图 14-1 (续)

466

```
        ObjectInputStream modelInObjectFile =
          new ObjectInputStream(new FileInputStream(modelName));
        messageCl = (MessageClassifier) modelInObjectFile.readObject();
        modelInObjectFile.close();
      } catch (FileNotFoundException e) {
        messageCl = new MessageClassifier();
      }

      // Check if there are any options left
      Utils.checkForRemainingOptions(options);

      // Process message.
      if (classValue.length() != 0) {
        messageCl.updateData(message.toString(), classValue);
      } else {
        messageCl.classifyMessage(message.toString());
      }

      // Save message classifier object.
      ObjectOutputStream modelOutObjectFile =
        new ObjectOutputStream(new FileOutputStream(modelName));
      modelOutObjectFile.writeObject(messageCl);
      modelOutObjectFile.close();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

图 14-1（续）

属性是通过调用类*Attribute*中的一个构造函数来完成的。在该类中有一个构造函数，该函数接受一个参数，即属性的名字，并生成一个数值性属性。然而，我们这里用的这个构造函数须接受两个参数：属性的名字和指向*FastVector*的一个参考。如果这个参考是空，就像该构造函数在我们的程序中的第一次应用那样，Weka会生成一个字符串类型的属性。否则的话，则生成一个名词性属性。我们在前面的探讨中是假定FastVector所持有的是字符串类型的属性值。这就是我们所谈到的用两个值hit和miss来生成一个类属性：即通过将属性的名字（class）和存储在一个*FastVector*中的该属性的值传给*Attribute()*。

要由属性信息生成一个数据集，*MessageClassifier()*必须生成一个*core*包中类*Instances*的对象。*MessageClassifier()*中调用的*Instances*类的构造函数接受三个可变参数：数据集的名字，含有数据集属性的一个*FastVector*，以及表示数据集初始容量的一个整数。我们将初始容量设为100；如果增加实例，其容量会自动增加。构建好数据集后，*MessageClassifier()*会把类属性的索引设定为最后一个属性的索引。 467

14.2.3 updateData()

现在用户知道了如何生成一个空的数据集，再来看*MessageClassifier*对象是如何具体收纳一个新的训练消息的。方法*updateData()*可完成这项工作。它首先调用*makeInstance()*来生成一个类*Instance*的对象，该对象对应着一个含有两个属性的实例，如此一来，它就把一个给定

的消息转换成了一个训练实例。*Instance*对象的构造函数将所有实例的值设为missing，其加权值设为1。*makeInstance()*接下来是设定持有消息文本的字符串属性的值。这可通过调用*Instance*对象中的*setValue()*方法来完成。有两个参数需要提供给这个方法，一个是其值需要被修改的属性，另一个参数是字符串属性的定义中对应着新值的索引。这个索引是由*addStringValue()*方法返回的，该方法将消息文本作为一个新值加到字符串属性中，并返回这个新值在字符串属性定义中的位置。

从内部机制来说，不论相对应的属性是什么类型的，一个*Instance*会把所有的属性值存储成双精度浮点小数。如果是名词性和字符串属性，只需存储相对应属性在属性定义的索引就可以了。例如，名词性属性的第一个值用0.0表示，第二个用1.0表示，并依此类推。同样方法也可用于字符串属性：*addStringValue()*返回的是对应于加到属性定义中的值的索引。

设定好字符串属性的值后，*makeInstance()*会将一个指向数据集的参考传给新生成的实例，从而使该实例可以读取数据的属性信息。在Weka中，一个*Instance*对象不会直接存储每个属性的类型；而是存储一个指向含有相应属性信息的数据集的一个参考。

现在回到*updateData()*。一旦*makeInstance()*返回了新的实例，它的类值就已经设定完成，且该实例已经被加进训练数据。我们还会将*m_UpToDate*初始化，它是一个显示训练数据已经发生变化，且预测模型因此需要更新的旗标。

14.2.4 classifyMessage()

现在让我们来看看*MessageClassifier*是如何处理一个类标签未知的消息的。*ClassifyMessage()*方法首先通过确定是否准备好训练数据，来查看一个分类器是否已经构建好。然后，再查看该分类器是否需要更新。如果分类器不是最新版本（因为训练数据已经被更改）它必须重新建立分类器。然而，在做这项工作以前，必须用*StringToWordVector*过滤器将数据转换成一个可适用于学习的格式。首先，我们用*setInputFormat()*将一个指向输入数据集的参考传递给该过滤器，以使其知道待输入数据的格式。每调用一次*setInputFormat()*方法，该过 468
滤器就初始化一遍。即所有的内部设置都被重新归位。下一步，用*useFilter()*对数据进行转换。这个来自于*Filter*类的通用方法将一个过滤器应用于一个数据集。此时，因为*StringToWordVector*刚刚被初始化，它会从训练数据集计算出一个字典，并用该字典形成一个字向量。从*useFilter()*方法返回后，所有该过滤器的内部设置都保持不变直到再一次调用*inputFormat()*令其重新初始化。这使得在无需更新过滤器的内部设置（在这种情况下指字典）的条件下过滤一个测试实例成为可能。

一旦数据被过滤，该程序就会将训练数据传递给它的方法*buildClassifier()*，重新构建分类器，在所举的例子中，就是J4.8决策树，并将*m_UpToDate*设为真（true）。在产生一个新的分类器之前，用*buildClassifier()*方法彻底初始化所用模型的内部设置是Weka的一个重要传统。因此，我们没有必要在调用*buildClassifier()*之前构建一个新的*J48*对象。

在确保存储在*m_Classifier*中的模型是最新的条件下，我们可以着手对消息进行分类了。在调用*makeInstance()*以便根据消息生成一个实例对象之前，需生成一个新的*Instances*对象用来持有这个实例，且该对象将作为一个可变参数传给*makeInstance()*。这样做的目的是使*makeInstance()*不必将消息文本加进*m_Data*中的字符串属性。否则，每当一个新消息被分类，*m_Data*对象的尺寸就会增大，这明显不合适，因为只有在加入训练实例的时候*m_Data*才应当

扩大。因此，生成一个临时的*Instances*对象，一旦它所持有的实例处理完毕，即将其删除。这个对象可用*stringFreeStructure()*方法得到，该方法返回一个含有空字符串属性的*m_Data*对象的复制。以上这些完成后，*makeInstance()*方法才会被调用以生成所要的新实例。

测试实例在分类之前也必须经*StringToWordVector*过滤器处理。这并不难：*input()*方法将实例送入过滤器对象，经过转化的实例可通过调用*output()*取得。然后，将实例传递给分类器的*classifyInstance()*方法，即可产生一个预测。正如用户所看到的，预测是作为一个双精度浮点小数编入代码的。这就使得Weka的评估模块可用类似的手段来处理类别化预测和数值性预测的模型。对于类别化预测，正如上面所举的例子，*double*变量的值就是所预测出的类值的索引。为了输出对应于这个类值的字符串，该程序调用了数据集类属性的*value()*方法。

至少有一种方式可以对我们以上的实现做出改进。分类器和*StringToWordVector*过滤器可以用第10.3节中描述过的*FilterClassifier*元学习器合并到一起。合并后的分类器便无需调用过滤器来转换数据，可直接处理字符串属性。我们没有这样做，是因为我们想演示过滤器是如

469 何通过编写代码的方式使用的。

第15章 编写新学习方案

如果用户需要实现一个Weka所没有的特殊目的的学习算法，或者用户正在进行机器学习的研究，并且想试验一个新的学习方案，或者用户只是想通过亲自动手编程，了解更多有关一个归纳算法的内部运作，本节用一个简单的范例演示在编写分类器时，如何充分利用Weka的类的层级结构，从而满足用户的需要。

Weka包含了表15-1中所列的基本的、主要用于教育目的的学习方案。表中的方案对于接受命令行选项没有特别要求。它们对于理解分类器的内部运作都很有用。我们会将*weka.classifiers.trees.Id3*作为一个例子讨论，该方案实现了第4.3节中的ID3决策树学习器。

表15-1 Weka中的简单学习方案

方　案	描　述	书中章节
weka.classifiers.bayes.NaiveBayesSimple	概率学习器	4.2
weka.classifiers.trees.Id3	决策树学习器	4.3
weka.classifiers.rules.Prism	规则学习器	4.4
weka.classifiers.lazy.IB1	基于实例的学习器	4.7

15.1 一个分类器范例

图15-1给出了*weka.classifiers.trees.Id3*的源代码，用户从代码中可看出它扩展*Classifier*类。无论是用于预测名词性类还是预测数值性类，每个Weka中的分类器都必须扩展*Classifier*类。

*weka.classifiers.trees.Id3*方案中的第一个方法是*globalInfo()*：我们在进入到更有趣的部分之前先谈谈这个方法。当这个方案在Weka的图形用户界面上被选中时，该方法只是简单地返回一个显示在屏幕上的字符串。

15.1.1 buildClassifier()

*buildClassifier()*方法根据训练数据集构建一个分类器。因为ID3算法无法处理非名词性类，残缺属性值、或任何非名词性的属性，因此，*buildClassifier()*方法首先在数据中对以上提到的进行查验。然后，它会生成一个训练集的复制件（以避免改变原始数据），并调用*weka.core.Instances*中的一个方法来删除所有含残缺类值的实例，因为这些实例在训练过程中不起作用。最后，它会调用*makeTree()*，该方法实际上通过递归的方式产生所有附加到根节点上的子树，从而生成一个决策树。

15.1.2 makeTree()

在*makeTree()*中，第一步是检查数据集是否为空。如果是，通过将*m_Attribute*设为空生成一个叶节点。为该叶指定的类值*m_ClassValue*设定为残缺，且*m_Distribution*中为数据集中的每个类所估计的概率皆初始化为0。如果训练实例已准备好，*makeTree()*会找出令这些实例产

生最大信息增益的属性。它首先生成一个数据集属性的Java枚举。如果类属性的索引已经设定，像正在讨论的这个数据集设定一样，该类属性会被自动排除在该枚举之外。

在枚举内部，每个属性的信息增益都由*computeInfoGain()*计算出来并存储在一个数组中。我们以后会重新讲这个方法。*weka.core.Attribute*中的*index()*方法可返回数据集中属性的索引。
471~472 它可为刚刚提到的数组编制索引。一旦完成了枚举，具有最大信息增益的属性就会存储在实例变量*m_Attribute*中。*Weka.core.Utils*中的*maxIndex()*方法会返回一个由整数或双精度浮点小数构成的数组中最大值的索引。（如果具有最大值的组元不止一个，那么只有第一个被返回。）该属性的索引会被传给*weka.core.Instances*中的*attribute()*方法，该方法返回与索引相对应的属性。

```
package weka.classifiers.trees;

import weka.classifiers.*;
import weka.core.*;
import java.io.*;
import java.util.*;

/**
 * Class implementing an Id3 decision tree classifier.
 */
public class Id3 extends Classifier {

  /** The node's successors. */
  private Id3[] m_Successors;

  /** Attribute used for splitting. */
  private Attribute m_Attribute;

  /** Class value if node is leaf. */
  private double m_ClassValue;

  /** Class distribution if node is leaf. */
  private double[] m_Distribution;

  /** Class attribute of dataset. */
  private Attribute m_ClassAttribute;

  /**
   * Returns a string describing the classifier.
   * @return a description suitable for the GUI.
   */
  public String globalInfo() {

    return  "Class for constructing an unpruned decision tree based on the ID3 "
      + "algorithm. Can only deal with nominal attributes. No missing values "
      + "allowed. Empty leaves may result in unclassified instances. For more "
      + "information see: \n\n"
      + " R. Quinlan (1986). \"Induction of decision "
      + "trees\". Machine Learning. Vol.1, No.1, pp. 81-106";
  }
```

473 图15-1 ID3决策树学习器的源代码

```
  /**
   * Builds Id3 decision tree classifier.
   *
   * @param data the training data
   * @exception Exception if classifier can't be built successfully
   */
  public void buildClassifier(Instances data) throws Exception {

    if (!data.classAttribute().isNominal()) {
      throw new UnsupportedClassTypeException("Id3: nominal class, please.");
    }
    Enumeration enumAtt = data.enumerateAttributes();
    while (enumAtt.hasMoreElements()) {
      if (!((Attribute) enumAtt.nextElement()).isNominal()) {
        throw new UnsupportedAttributeTypeException("Id3: only nominal " +
                                                    "attributes, please.");
      }
    }
    Enumeration enum = data.enumerateInstances();
    while (enum.hasMoreElements()) {
      if (((Instance) enum.nextElement()).hasMissingValue()) {
        throw new NoSupportForMissingValuesException("Id3: no missing values, "
                                                     + "please.");

      }
    }
    data = new Instances(data);
    data.deleteWithMissingClass();
    makeTree(data);
  }

  /**
   * Method for building an Id3 tree.
   *
   * @param data the training data
   * @exception Exception if decision tree can't be built successfully
   */
  private void makeTree(Instances data) throws Exception {

    // Check if no instances have reached this node.
    if (data.numInstances() == 0) {
      m_Attribute = null;
    m_ClassValue = Instance.missingValue();
    m_Distribution = new double[data.numClasses()];
    return;
  }

  // Compute attribute with maximum information gain.
  double[] infoGains = new double[data.numAttributes()];
  Enumeration attEnum = data.enumerateAttributes();
  while (attEnum.hasMoreElements()) {
    Attribute att = (Attribute) attEnum.nextElement();
```

474

图 15-1 (续)

```
      infoGains[att.index()] = computeInfoGain(data, att);
    }
    m_Attribute = data.attribute(Utils.maxIndex(infoGains));

    // Make leaf if information gain is zero.
    // Otherwise create successors.
    if (Utils.eq(infoGains[m_Attribute.index()], 0)) {
      m_Attribute = null;
      m_Distribution = new double[data.numClasses()];
      Enumeration instEnum = data.enumerateInstances();
      while (instEnum.hasMoreElements()) {
        Instance inst = (Instance) instEnum.nextElement();
        m_Distribution[(int) inst.classValue()]++;
      }
      Utils.normalize(m_Distribution);
      m_ClassValue = Utils.maxIndex(m_Distribution);
      m_ClassAttribute = data.classAttribute();
    } else {
      Instances[] splitData = splitData(data, m_Attribute);
      m_Successors = new Id3[m_Attribute.numValues()];
      for (int j = 0; j < m_Attribute.numValues(); j++) {
        m_Successors[j] = new Id3();
        m_Successors[j].makeTree(splitData[j]);
      }
    }
  }

  /**
   * Classifies a given test instance using the decision tree.
   *
475  * @param instance the instance to be classified
   * @return the classification
   *
  public double classifyInstance(Instance instance)
    throws NoSupportForMissingValuesException {

    if (instance.hasMissingValue()) {
      throw new NoSupportForMissingValuesException("Id3: no missing values, "
                                                   + "please.");
    }
    if (m_Attribute == null) {
      return m_ClassValue;
    } else {
      return m_Successors[(int) instance.value(m_Attribute)].
        classifyInstance(instance);
    }
  }

   **
   * Computes class distribution for instance using decision tree.
   *
   * @param instance the instance for which distribution is to be computed
```

图 15-1（续）

```
   * @return the class distribution for the given instance
   */
  public double[] distributionForInstance(Instance instance)
    throws NoSupportForMissingValuesException {

    if (instance.hasMissingValue()) {
      throw new NoSupportForMissingValuesException("Id3: no missing values, "
                                                   + "please.");
    }
    if (m_Attribute == null) {
      return m_Distribution;
    } else {
      return m_Successors[(int) instance.value(m_Attribute)].
        distributionForInstance(instance);
    }
  }

  /**
   * Prints the decision tree using the private toString method from below.
   *
   * @return a textual description of the classifier
   */
  public String toString() {

    if ((m_Distribution == null) && (m_Successors == null)) {
      return "Id3: No model built yet.";
    }
    return "Id3\n\n" + toString(0);
  }

  /**
   * Computes information gain for an attribute.
   *
   * @param data the data for which info gain is to be computed
   * @param att the attribute
   * @return the information gain for the given attribute and data
   */
  private double computeInfoGain(Instances data, Attribute att)
    throws Exception {

    double infoGain = computeEntropy(data);
    Instances[] splitData = splitData(data, att);
    for (int j = 0; j < att.numValues(); j++) {
      if (splitData[j].numInstances() > 0) {
        infoGain -= ((double) splitData[j].numInstances() /
                     (double) data.numInstances()) *
          computeEntropy(splitData[j]);
      }
    }
    return infoGain;
  }

  /**
```

476

图 15-1 (续)

```
 * Computes the entropy of a dataset.
 *
 * @param data the data for which entropy is to be computed
 * @return the entropy of the data's class distribution
 */
private double computeEntropy(Instances data) throws Exception {

477  double [] classCounts = new double[data.numClasses()];
  Enumeration instEnum = data.enumerateInstances();
  while (instEnum.hasMoreElements()) {
    Instance inst = (Instance) instEnum.nextElement();
    classCounts[(int) inst.classValue()]++;
  }
  double entropy = 0;
  for (int j = 0; j < data.numClasses(); j++) {
    if (classCounts[j] > 0) {
      entropy -= classCounts[j] * Utils.log2(classCounts[j]);
    }
  }
  entropy /= (double) data.numInstances();
  return entropy + Utils.log2(data.numInstances());
}

/**
 * Splits a dataset according to the values of a nominal attribute.
 *
 * @param data the data which is to be split
 * @param att the attribute to be used for splitting
 * @return the sets of instances produced by the split
 */
private Instances[] splitData(Instances data, Attribute att) {

  Instances[] splitData = new Instances[att.numValues()];
  for (int j = 0; j < att.numValues(); j++) {
    splitData[j] = new Instances(data, data.numInstances());
  }
  Enumeration instEnum = data.enumerateInstances();
  while (instEnum.hasMoreElements()) {
    Instance inst = (Instance) instEnum.nextElement();
    splitData[(int) inst.value(att)].add(inst);
  }
  for (int i = 0; i < splitData.length; i++) {
    splitData[i].compactify();
  }
  return splitData;
}

/**
478 * Outputs a tree at a certain level.
 *
 * @param level the level at which the tree is to be printed
 */
private String toString(int level) {
```

图 15-1（续）

```
    StringBuffer text = new StringBuffer();

    if (m_Attribute == null) {
      if (Instance.isMissingValue(m_ClassValue)) {
        text.append(": null");
      } else {
        text.append(": " + m_ClassAttribute.value((int) m_ClassValue));
      }
    } else {
      for (int j = 0; j < m_Attribute.numValues(); j++) {
        text.append("\n");
        for (int i = 0; i < level; i++) {
          text.append("|  ");
        }
        text.append(m_Attribute.name() + " = " + m_Attribute.value(j));
        text.append(m_Successors[j].toString(level + 1));
      }
    }
    return text.toString();
  }

  /**
   * Main method.
   *
   * @param args the options for the classifier
   */
  public static void main(String[] args) {

    try {
      System.out.println(Evaluation.evaluateModel(new Id3(), args));
    } catch (Exception e) {
      System.err.println(e.getMessage());
    }
  }
}
```

图 15-1（续） 479

用户也许在想，数组中与类属性相对应的那个值域怎么样了？这个不必担心，因为Java会自动将数组中所有组元初始化为整数0，而信息增益总是大于或等于0。如果最大信息增益是0，*makeTree()*会生成一个叶节点。在这种情况下，*makeTree()*会设为空，且*makeTree()*会同时计算类概率的分布以及具有最大概率的类。（*weka.core.Utils*中的*normalize()*方法会将一个双精度浮点小数数组正常化使其组员相加总和为1。）

当它产生一个已指定类值的叶节点时，*makeTree()*将类属性存储到*m_ClassAttribute*中。这是因为用来输出决策树的方法需要读取该类值以便显示类标签。

如果发现了一个具有非零信息增益的属性，*makeTree()*会根据该属性的值分割数据集，并以递归的方式为每个新产生的数据集构建子树。该方法调用另一个方法*splitData()*进行分割。这样就会生成与属性值一样多个空的数据集，且把这些数据集存储到一个数组中（将每个数据集的初始容量设定为原始数据集中所含实例的数量），然后在原始数据集中将每个实例依次循环一遍，并在新数据集中根据相对应的属性值为这些实例开辟空间。然后压缩*Instances*对

象以减少占用的存储器。返回到*makeTree()*后，所得到的数据集数组用于构建子树。该方法会生成一个由*Id3*对象构成的数组，数组中的每个对象对应着一个属性值，并将相对应的数据集传给*makeTree()*，从而在每个对象上调用该方法。

15.1.3 computeInfoGain()

现在回到*computeInfoGain()*，与一个属性和一个数据集相关联的信息增益是用第4.3节中介绍过的方程式的一个直接实现计算出来的。首先计算出数据集的熵，然后用*splitData()*将数据集分割成子集，并在每个子集上调用*computeEntropy()*。最后，将前面计算出来的熵与后面计算出来的每个熵的加权总和相减的差，即信息增益返回。*computeEntropy()*方法使用*weka.core.Utils*中的*log2()*方法得出一个数的对数（以2为基数）。

15.1.4 classifyInstance()

看过了ID3如何构建决策树，我们再来看如何利用树结构来预测类值及概率。每一个分类器都必须实现*classifyInstance()*方法或*distributionForInstance()*方法（或两个方法都实现）。
480 *Classifier*超类含有这两种方法的默认实现。*classifyInstance()*的默认实现调用*distributionForInstance()*。如果类是名词性的，*classifyInstance()*会把具有最大概率的属性预测为类，否则，如果从*distributionForInstance()*返回的所有概率都是零，*classifyInstance()*会返回一个残缺值。如果类是数值性的，*distributionForInstance()*必须返回有数值性预测的单一组元数组，该数组也就是*classifyInstance()*要提取并返回的。最后，*distributionForInstance()*的默认实现反过来把从*classifyInstance()*中得来的预测包装成一个单一组元数组。如果类是名词性的，*distributionForInstance()*将概率1指定给*classifyInstance()*预测出的类属性，把概率0指定给其他属性。如果*classifyInstance()*返回一个残缺值，所有属性的概率都设为0。为了让用户更好地了解这些方法所做的工作、*weka.classifiers.trees.Id3*类重新编写了这两个方法。

我们先来看看针对一个给定实例预测类值的*classifyInstance()*。上一节曾经讲过，与名词性属性值一样，名词性类值是以*double*变量的形式编码及存储的，表示值的名字在属性声明中的索引。这种更简洁有效的面向对象的处理方式可加快运行速度。在ID3的实现中，*classifyInstance()*首先查看待分类的实例中是否有残缺值。如果有的话，就丢弃一个异常。否则，它就以递归的方式，根据待分类实例的属性值，沿着树自上而下，直至到达某个末端叶节点。然后，它会返回存储在该叶节点的类值*m_ClassValue*。要注意所返回的也有可能是残缺值，如果是残缺值，该实例则成为未被分类的实例。*distributionForInstance()*方法的工作方式与此完全一样，它返回存储于*m_Distribution*中的概率分布。

大多数机器学习模型，特别是决策树，大致上全面反映了数据本身的结构。因此每个Weka分类器，如同许多其他Java对象一样，实现*toString()*方法从而以字符串变量的形式生成一个它自身的文本表述。ID3的*toString()*方法输出一个与J4.8格式大致相同的决策树（图10-5）。它通过读取存储于节点上的属性信息，以递归的方式将树的结构输入一个字符串变量。它使用*weka.core.Attribute*中的*name()*和*value()*方法得到每个属性的名字和值。不含类值的空末端叶节点由字符串*null*标示出来。

15.1.5 main()

*weka.classifiers.tree.Id3*中还没有被描述的唯一方法就是*main()*，每当由命令行执行一个类，

该方法都会被调用。正如用户看到的一样，该方法很简单：基本上就是告诉Weka的类*Evaluation*用所给的命令行选项评估*Id3*，并输出所得到的字符串。完成此项任务的单行表达式就包含在一个*try-catch*语句中，该语句能捕获各种各样由Weka例程或其他Java方法丢出的异常。 481

*Weka.classifiers.Evaluation*中的*evaluation()*方法解释了第13.3节中讨论过的，可适用于任何学习方案的通用命令行选项及相应的作用。例如，它可接受训练文件名字的-t选项，并载入相对应的数据集。如果没有测试文件，它就进行交叉验证，方式是生成一个分类器，并在训练数据的不同的子集上重复调用*buildClassifier()*、*classifyInstance()*和*distributionForInstance()*。除非用户设定了相应的命令行选项从而阻止模型的输出，它还会调用*toString()*方法，输出由整个训练数据集生成的模型。

如果某个学习方案需要解释一个具体的选项，比如一个修剪参数，怎么办？这可由*weka.core*中的*OptionHandler*接口来解决。实现该接口的分类器含有三个方法：*listOptions()*、*setOptions()*和*getOptions()*。它们分别用来列出所有针对该分类器的选项，设定其中某些选项，以及取得目前已设定的选项。如果一个分类器实现了*OptionHandler*接口，*Evaluation*类中的*evaluation()*方法会自动调用这些方法。处理完通用选项后，*evaluation()*会调用*setOption()*来处理余下的选项，然后利用*buildClassifier()*产生一个新的分类器。输出所产生的分类器，*evaluation()*会用*getOptions()*输出一列目前已设定的选项。在*weka.classifiers.rules.OneR*的源代码中可找到一个如何实现这些方法的简单范例。

*OptionHandler*使得在命令行中设定选项成为可能。要在图形用户界面中设定这些选项，Weka使用的是Java豆的架构。实施该构架所要做的全部工作就是为一个类中所用到的每个参数都提供*set...()*及*get...()*方法。比方说，方法*setPruningParameter()*和*getPruningParameter()*对于一个修剪参数来说就是必须的。还有一个方法也必不可少，*pruningParameterTipText()*返回的是显示在图形用户界面上的对该参数的一个描述。再强调一次，见*weka.classifiers.rules.OneR*中的例子。

一些分类器可在新的训练实例陆续到达时进行递增更新，并且不需要在同一批中处理全部数据。在Weka中，递增分类器须实现*weka.classifiers*中的*UpdateableClassifier*接口。该接口只声明了一个名为*updateClassifier()*的方法，该方法只接受一个单独的训练实例作为它的可变参数。要参考一个如何使用该接口的例子，见*weka.classifiers.lazy.IBk*的源代码。

如果一个分类器能运用实例的权，它必须实现*weka.core*中的*WeightedInstancesHandler()*接口。如此一来其他的算法，比方说那些用于提升的算法，就可对该属性加以利用。

在*weka.core*中还有很多其他对于分类器来说很有用的接口，例如，*randomizable*、*summarizable*、*drawable*、和*graphable*这些用于分类器的接口。有关接口的更多信息，见*weka.core*中相应类的Javadoc。 482

15.2 与实现分类器有关的惯例

在实现*Weka*中的分类器时，有一些惯例用户必须遵守。否则，程序会出错。比方说，Weka的评估模块在评估分类器时可能会无法恰当地计算它的统计数据。

第一个惯例前面已经提到过，当一个分类器的*buildClassifier()*方法被调用时，必须令模型重新复位。类*CheckClassifier*进行测试，确保模型的确被复位了。当*buildClassifier()*在某个数据集上被调用时，无论该分类器以前已经在同一个或其他的数据集上被调用过多少次，所

得到的结果必须是一样的。还有，一些实例变量是与某些只适用于具体方案的选项相对应的，*buildClassifier()*方法绝对不可以将这些变量复位，因为这些变量的值一旦被设定，它们在多次调用*buildClassifier()*的过程中必须保持不变。还有，调用*buildClassifier()*绝对不可以改动输入数据。

另外两个惯例以前也提到过。一个是当某个分类器无法做出预测时，它的*classifyInstance()*方法必须返回*Instance.missingValue()*，且它的*distributionForInstance()*方法必须针对所有类属性都返回0概率。图15-1中的ID3实现就是这么做的。另外一个惯例是这样的，对于用作数值性预测的分类器来说，它的*classifyInstance()*要返回分类器所预测出的数值性类值。还有一些分类器可以对名词性的类和类概率，以及数值性的类值做出预测，*weka.classifiers.lazy.IBk*就是一个例子。这些分类器实现了*distributionForInstance()*方法，如果类是数值性的，它会返回一个单一组元数组，其唯一组元就含有所预测的数值性值。

另外一个惯例虽然并不是不可或缺，但不管怎么说都是有益的，即每个分类器都实现一个*toString()*方法，用于输出一个它自身的文本描述。 483

参 考 文 献

Adriaans, P., and D. Zantige. 1996. *Data mining.* Harlow, England: Addison-Wesley.

Agrawal, R., and R. Srikant. 1994. Fast algorithms for mining association rules in large databases. In J. Bocca, M. Jarke, and C. Zaniolo, editors, *Proceedings of the International Conference on Very Large Databases,* Santiago, Chile. San Francisco: Morgan Kaufmann, pp. 478–499.

Agrawal, R., T. Imielinski, and A. Swami. 1993a. Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engineering* 5(6): 914–925.

———. 1993b. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Washington, DC. New York: ACM, pp. 207–216.

Aha, D. 1992. Tolerating noisy, irrelevant, and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies* 36(2):267–287.

Almuallin, H., and T. G. Dietterich. 1991. Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence,* Anaheim, CA. Menlo Park, CA: AAAI Press, pp. 547–552.

———. 1992. Efficient algorithms for identifying relevant features. In *Proceedings of the Ninth Canadian Conference on Artificial Intelligence,* Vancouver, BC. San Francisco: Morgan Kaufmann, pp. 38–45.

Appelt, D. E. 1999. Introduction to information extraction technology. *Tutorial, Int Joint Conf on Artificial Intelligence IJCAI'99.* Morgan Kaufmann, San Mateo. Tutorial notes available at www.ai.sri.com/~appelt/ie-tutorial.

Asmis, E. 1984. *Epicurus' scientific method.* Ithaca, NY: Cornell University Press.

Atkeson, C. G., S. A. Schaal, and A. W. Moore. 1997. Locally weighted learning. *AI Review* 11:11–71.

Bay, S. D. 1999. Nearest-neighbor classification from multiple feature subsets. *Intelligent Data Analysis* 3(3):191–209.

Bay, S. D., and M. Schwabacher. 2003. Near linear time detection of distance-based outliers and applications to security. In *Proceedings of the Workshop on Data Mining for Counter Terrorism and Security,* San Francisco. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Bayes, T. 1763. An essay towards solving a problem in the doctrine of chances.

Philosophical Transactions of the Royal Society of London 53:370–418.

Beck, J. R., and E. K. Schultz. 1986. The use of ROC curves in test performance evaluation. *Archives of Pathology and Laboratory Medicine* 110:13–20.

Bergadano, F., and D. Gunetti. 1996. *Inductive logic programming: From machine learning to software engineering.* Cambridge, MA: MIT Press.

Berners-Lee, T., J. Hendler, and O. Lassila. 2001. The semantic web. *Scientific American* 284(5):34–43.

Berry, M. J. A., and G. Linoff. 1997. *Data mining techniques for marketing, sales, and customer support.* New York: John Wiley.

Bigus, J. P. 1996. *Data mining with neural networks.* New York: McGraw Hill.

Bishop, C. M. 1995. *Neural networks for pattern recognition.* New York: Oxford University Press.

Blake, C., E. Keogh, and C. J. Merz. 1998. *UCI Repository of machine learning databases* [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Department of Information and Computer Science, University of California, Irvine, CA.

BLI (Bureau of Labour Information). 1988. *Collective Bargaining Review* (November). Ottawa, Ontario, Canada: Labour Canada, Bureau of Labour Information.

Blum, A., and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, Madison, WI. San Francisco: Morgan Kaufmann, pp. 92–100.

Bouckaert, R. R. 1995. *Bayesian belief networks: From construction to inference.* PhD Dissertation, Computer Science Department, University of Utrecht, The Netherlands.

Bouckaert, R. R. 2004. Bayesian network classifiers in Weka. Working Paper 14/2004, Department of Computer Science, University of Waikato, New Zealand.

Brachman, R. J., and H. J. Levesque, editors. 1985. *Readings in knowledge representation.* San Francisco: Morgan Kaufmann.

Brefeld, U., and T. Scheffer. 2004. Co-EM support vector learning. In R. Greiner and D. Schuurmans, editors, *Proceedings of the Twenty-First International Conference on Machine Learning*, Banff, Alberta, Canada. New York: ACM, pp. 121–128.

Breiman, L. 1996a. Stacked regression. *Machine Learning* 24(1):49–64.

———. 1996b. Bagging predictors. *Machine Learning* 24(2):123–140.

———. 1999. Pasting small votes for classification in large databases and online. *Machine Learning* 36(1–2):85–103.

———. 2001. Random forests. *Machine Learning* 45(1):5–32.

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and regression trees.* Monterey, CA: Wadsworth.

Brin, S., R. Motwani, J. D. Ullman, and S. Tsur. 1997. Dynamic itemset counting and implication rules for market basket data. *ACM SIGMOD Record* 26(2):255–264.

Brodley, C. E., and M. A. Friedl. 1996. Identifying and eliminating mislabeled training instances. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, OR. Menlo Park, CA: AAAI Press, pp. 799–805.

Brownstown, L., R. Farrell, E. Kant, and N. Martin. 1985. *Programming expert systems in OPS5.* Reading, MA: Addison-Wesley.

Buntine, W. 1992. Learning classification trees. *Statistics and Computing* 2(2):63–73.

Burges, C. J. C. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery* 2(2): 121–167.

Cabena, P., P. Hadjinian, R. Stadler, J. Verhees, and A. Zanasi. 1998. *Discovering data mining: From concept to implementation.* Upper Saddle River, NJ: Prentice Hall.

Califf, M. E., and R. J. Mooney. 1999. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, Orlando, FL. Menlo Park, AC: AAAI Press, pp. 328–334.

Cardie, C. 1993. Using decision trees to improve case-based learning. In P. Utgoff, editor, *Proceedings of the Tenth International Conference on Machine Learning*, Amherst, MA. San Francisco: Morgan Kaufmann, pp. 25–32.

Cavnar, W. B., and J. M. Trenkle. 1994. N-Gram-based text categorization. *Proceedings of the Third Symposium on Document Analysis and Information Retrieval.* Las Vegas, NV, UNLV Publications/Reprographics, pp. 161–175.

Cendrowska, J. 1998. PRISM: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies* 27(4):349–370.

Chakrabarti, S. 2003. *Mining the web: discovering knowledge from hypertext data.* San Francisco, CA: Morgan Kaufmann.

Cheeseman, P., and J. Stutz. 1995. Bayesian classification (AutoClass): Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining.* Menlo Park, CA: AAAI Press, pp. 153–180.

Chen, M.S., J. Jan, and P. S. Yu. 1996. Data mining: An overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering* 8(6): 866–883.

Cherkauer, K. J., and J. W. Shavlik. 1996. Growing simpler decision trees to facilitate knowledge discovery. In E. Simoudis, J. W. Han, and U. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Portland, OR. Menlo Park, CA: AAAI Press, pp. 315–318.

Cleary, J. G., and L. E. Trigg. 1995. K*: An instance-based learner using an entropic distance measure. In A. Prieditis and S. Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, Tahoe City, CA. San Francisco: Morgan Kaufmann, pp. 108–114.

Cohen, J. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 20:37–46.

Cohen, W. W. 1995. Fast effective rule induction. In A. Prieditis and S. Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, Tahoe City, CA. San Francisco: Morgan Kaufmann, pp. 115–123.

Cooper, G. F., and E. Herskovits. 1992. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning* 9(4):309–347.

Cortes, C., and V. Vapnik. 1995. Support vector networks. *Machine Learning* 20(3):273–297.

Cover, T. M., and P. E. Hart. 1967. Nearest-neighbor pattern classification. *IEEE Transactions on Information Theory* IT-13:21–27.

Cristianini, N., and J. Shawe-Taylor. 2000. *An introduction to support vector machines and other kernel-based learning methods.* Cambridge, UK: Cambridge University Press.

Cypher, A., editor. 1993. *Watch what I do: Programming by demonstration.* Cambridge, MA: MIT Press.

Dasgupta, S. 2002. Performance guarantees for hierarchical clustering. In J. Kivinen and R. H. Sloan, editors, *Proceedings of the Fifteenth Annual Conference on Computational Learning Theory*, Sydney, Australia. Berlin: Springer-Verlag, pp. 351–363.

Datta, S., H. Kargupta, and K. Sivakumar. 2003. Homeland defense, privacy-sensitive data mining, and random value distortion. In *Proceedings of the Workshop on Data Mining for Counter Terrorism and Security*, San Francisco. Society for International and Applied Mathematics, Philadelphia, PA.

Demiroz, G., and A. Guvenir. 1997. Classification by voting feature intervals. In M. van Someren and G. Widmer, editors, *Proceedings of the Ninth European Conference on Machine Learning*, Prague, Czech Republic. Berlin: Springer-Verlag, pp. 85–92.

Devroye, L., L. Györfi, and G. Lugosi. 1996. *A probabilistic theory of pattern recognition.* New York: Springer-Verlag.

Dhar, V., and R. Stein. 1997. *Seven methods for transforming corporate data into business intelligence.* Upper Saddle River, NJ: Prentice Hall.

Diederich, J., J. Kindermann, E. Leopold, and G. Paass. 2003. Authorship attribution with support vector machines. *Applied Intelligence* 19(1):109–123.

Dietterich, T. G. 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization.

Machine Learning 40(2):139–158.

Dietterich, T. G., and G. Bakiri. 1995. Solving multiclass learning problems via error-correcting output codes. *Journal Artificial Intelligence Research* 2:263–286.

Domingos, P. 1997. Knowledge acquisition from examples via multiple models. In D. H. Fisher Jr., editor, *Proceedings of the Fourteenth International Conference on Machine Learning*, Nashville, TN. San Francisco: Morgan Kaufmann, pp. 98–106.

———. 1999. MetaCost: A general method for making classifiers cost sensitive. In U. M. Fayyad, S. Chaudhuri, and D. Madigan, editors, *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, San Diego, CA. New York: ACM, pp. 155–164.

Dougherty, J., R. Kohavi, and M. Sahami. 1995. Supervised and unsupervised discretization of continuous features. In A. Prieditis and S. Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, Tahoe City, CA. San Francisco: Morgan Kaufmann, pp. 194–202.

Drucker, H. 1997. Improving regressors using boosting techniques. In D. H. Fisher, editor, *Proceedings of the Fourteenth International Conference on Machine Learning*, Nashville, TN. San Francisco: Morgan Kaufmann, pp. 107–115.

Drummond, C., and R. C. Holte. 2000. Explicitly representing expected cost: An alternative to ROC representation. In R. Ramakrishnan, S. Stolfo, R. Bayardo, and I. Parsa, editors, *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining*, Boston, MA. New York: ACM, pp. 198–207.

Duda, R. O., and P. E. Hart. 1973. *Pattern classification and scene analysis*. New York: John Wiley.

Duda, R. O., P. E. Hart, and D. G. Stork. 2001. *Pattern Classification*, second edition. New York: John Wiley.

Dumais, S. T., J. Platt, D. Heckerman, and M. Sahami. 1998. Inductive learning algorithms and representations for text categorization. In *Proceedings of the ACM Seventh International Conference on Information and Knowledge Management*, Bethesda, MD. New York: ACM, pp. 148–155.

Efron, B., and R. Tibshirani. 1993. *An introduction to the bootstrap*. London: Chapman and Hall.

Egan, J. P. 1975. *Signal detection theory and ROC analysis*. Series in Cognition and Perception. New York: Academic Press.

Fayyad, U. M., and K. B. Irani. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, Chambery, France. San Francisco: Morgan Kaufmann, pp. 1022–1027.

Fayyad, U. M., and P. Smyth. 1995. From massive datasets to science catalogs:

Applications and challenges. In *Proceedings of the Workshop on Massive Datasets.* Washington, DC: NRC, Committee on Applied and Theoretical Statistics.

Fayyad, U. M., G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. 1996. *Advances in knowledge discovery and data mining.* Menlo Park, CA: AAAI Press/MIT Press.

Fisher, D. 1987. Knowledge acquisition via incremental conceptual clustering. *Machine Learning* 2(2):139–172.

Fisher, R. A. 1936. The use of multiple measurements in taxonomic problems. *Annual Eugenics* 7(part II):179–188. Reprinted in *Contributions to Mathematical Statistics*, 1950. New York: John Wiley.

Fix, E., and J. L. Hodges Jr. 1951. Discriminatory analysis; nonparametric discrimination: Consistency properties. Technical Report 21-49-004(4), USAF School of Aviation Medicine, Randolph Field, Texas.

Flach, P. A., and N. Lachiche. 1999. Confirmation-guided discovery of first-order rules with Tertius. *Machine Learning* 42:61–95.

Fletcher, R. 1987. *Practical methods of optimization*, second edition. New York: John Wiley.

Fradkin, D., and D. Madigan. 2003. Experiments with random projections for machine learning. In L. Getoor, T. E. Senator, P. Domingos, and C. Faloutsos, editors, *Proceedings of the Ninth International Conference on Knowledge Discovery and Data Mining*, Washington, DC. New York: ACM, pp. 517–522.

Frank E. 2000. *Pruning decision trees and lists.* PhD Dissertation, Department of Computer Science, University of Waikato, New Zealand.

Frank, E., and M. Hall. 2001. A simple approach to ordinal classification. In L. de Raedt and P. A. Flach, editors, *Proceedings of the Twelfth European Conference on Machine Learning*, Freiburg, Germany. Berlin: Springer-Verlag, pp. 145–156.

Frank, E., and I. H. Witten. 1998. Generating accurate rule sets without global optimization. In J. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning*, Madison, WI. San Francisco: Morgan Kaufmann, pp. 144–151.

———. 1999. Making better use of global discretization. In I. Bratko and S. Dzeroski, editors, *Proceedings of the Sixteenth International Conference on Machine Learning*, Bled, Slovenia. San Francisco: Morgan Kaufmann, pp. 115–123.

Frank, E., M. Hall, and B. Pfahringer. 2003. Locally weighted Naïve Bayes. In U. Kjærulff and C. Meek, editors, *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, Acapulco, Mexico. San Francisco: Morgan Kaufmann, pp. 249–256.

Frank, E., G. Holmes, R. Kirkby, and M. Hall. 2002. Racing committees for large datasets. In S. Lange and K. Satoh, and C. H. Smith, editors, *Proceedings of the Fifth International Conference on Discovery Science*, Lübeck, Germany. Berlin:

Springer-Verlag, pp. 153–164.

Frank, E., G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. 1999. Domain-specific key phrase extraction. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, Stockholm, Sweden. San Francisco: Morgan Kaufmann, pp. 668–673.

Freitag, D. 2002. Machine learning for information extraction in informal domains. *Machine Learning* 39(2/3):169–202.

Freund, Y., and L. Mason. 1999. The alternating decision-tree learning algorithm. In I. Bratko and S. Dzeroski, editors, *Proceedings of the Sixteenth International Conference on Machine Learning*, Bled, Slovenia. San Francisco: Morgan Kaufmann, pp. 124–133.

Freund, Y., and R. E. Schapire. 1996. Experiments with a new boosting algorithm. In L. Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, Bari, Italy. San Francisco: Morgan Kaufmann, pp. 148–156.

———. 1999. Large margin classification using the perceptron algorithm. *Machine Learning* 37(3):277–296.

Friedman, J. H. 1996. Another approach to polychotomous classification. Technical Report, Department of Statistics, Stanford University, Stanford, CA.

———. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics* 29(5):1189–1232.

Friedman, J. H., J. L. Bentley, and R. A. Finkel. 1977. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software* 3(3):209–266.

Friedman, J. H., T. Hastie, and R. Tibshirani. 2000. Additive logistic regression: A statistical view of boosting. *Annals of Statistics* 28(2):337–374.

Friedman, N., D. Geiger, and M. Goldszmidt. 1997. Bayesian network classifiers. *Machine Learning* 29(2):131–163.

Fulton, T., S. Kasif, and S. Salzberg. 1995. Efficient algorithms for finding multiway splits for decision trees. In A. Prieditis and S. Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, Tahoe City, CA. San Francisco: Morgan Kaufmann, pp. 244–251.

Fürnkrantz, J. 2002. Round robin classification. *Journal of Machine Learning Research* 2:721–747.

Fürnkrantz, J., and P. A. Flach. 2005. ROC 'n' rule learning: Towards a better understanding of covering algorithms. *Machine Learning* 58(1):39–77.

Fürnkrantz, J., and G. Widmer. 1994. Incremental reduced-error pruning. In H. Hirsh and W. Cohen, editors, *Proceedings of the Eleventh International Conference on Machine Learning*, New Brunswick, NJ. San Francisco: Morgan Kaufmann, pp. 70–77.

Gaines, B. R., and P. Compton. 1995. Induction of ripple-down rules applied to

modeling large databases. *Journal of Intelligent Information Systems* 5(3):211–228.

Genesereth, M. R., and N. J. Nilsson. 1987. *Logical foundations of artificial intelligence.* San Francisco: Morgan Kaufmann.

Gennari, J. H., P. Langley, and D. Fisher. 1990. Models of incremental concept formation. *Artificial Intelligence* 40:11–61.

Ghani, R. 2002. Combining labeled and unlabeled data for multiclass text categorization. In C. Sammut and A. Hoffmann, editors, *Proceedings of the Nineteenth International Conference on Machine Learning,* Sydney, Australia. San Francisco: Morgan Kaufmann, pp. 187–194.

Gilad-Bachrach, R., A. Navot, and N. Tishby. 2004. Margin based feature selection: Theory and algorithms. In R. Greiner and D. Schuurmans, editors, *Proceedings of the Twenty-First International Conference on Machine Learning,* Banff, Alberta, Canada. New York: ACM, pp. 337–344.

Giraud-Carrier, C. 1996. FLARE: Induction with prior knowledge. In J. Nealon and J. Hunt, editors, *Research and Development in Expert Systems XIII.* Cambridge, UK: SGES Publications, pp. 11–24.

Gluck, M., and J. Corter. 1985. Information, uncertainty, and the utility of categories. In *Proceedings of the Annual Conference of the Cognitive Science Society,* Irvine, CA. Hillsdale, NJ: Lawrence Erlbaum, pp. 283–287.

Goldberg, D. E. 1989. *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley.

Good, P. 1994. *Permutation tests: A practical guide to resampling methods for testing hypotheses.* Springer-Verlag, New York, NY.

Grossman, D., and P. Domingos. 2004. Learning Bayesian network classifiers by maximizing conditional likelihood. In R. Greiner and D. Schuurmans, editors, *Proceedings of the Twenty-First International Conference on Machine Learning,* Banff, Alberta, Canada. New York: ACM, pp. 361–368.

Groth, R. 1998. *Data mining: A hands-on approach for business professionals.* Upper Saddle River, NJ: Prentice Hall.

Guo, Y., and R. Greiner. 2004. Discriminative model selection for belief net structures. Department of Computing Science, TR04-22, University of Alberta, Canada.

Guyon, I., J. Weston, S. Barnhill, and V. Vapnik. 2002. Gene selection for cancer classification using support vector machines. *Machine Learning* 46(1–3): 389–422.

Hall, M. 2000. Correlation-based feature selection for discrete and numeric class machine learning. In P. Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning,* Stanford, CA. San Francisco: Morgan Kaufmann, pp. 359–366.

Hall, M., G. Holmes, and E. Frank. 1999. Generating rule sets from model trees. In N. Y. Foo, editor, *Proceedings of the Twelfth Australian Joint Conference on Artificial Intelligence*, Sydney, Australia. Berlin: Springer-Verlag, pp. 1–12.

Han, J., and M. Kamber. 2001. *Data mining: Concepts and techniques.* San Francisco: Morgan Kaufmann.

Hand, D. J., H. Mannila, and P. Smyth. 2001. *Principles of data mining.* Cambridge, MA: MIT Press.

Hartigan, J. A. 1975. *Clustering algorithms.* New York: John Wiley.

Hastie, T., and R. Tibshirani. 1998. Classification by pairwise coupling. *Annals of Statistics* 26(2):451–471.

Hastie, T., R. Tibshirani, and J. Friedman. 2001. *The elements of statistical learning.* New York: Springer-Verlag.

Heckerman, D., D. Geiger, and D. M. Chickering. 1995. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning* 20(3):197–243.

Hochbaum, D. S., and D. B. Shmoys. 1985. A best possible heuristic for the *k*-center problem. *Mathematics of Operations Research* 10(2):180–184.

Holmes, G., and C. G. Nevill-Manning. 1995. Feature selection via the discovery of simple classification rules. In G. E. Lasker and X. Liu, editors, *Proceedings of the International Symposium on Intelligent Data Analysis.* Baden-Baden, Germany: International Institute for Advanced Studies in Systems Research and Cybernetics, pp. 75–79.

Holmes, G., B. Pfahringer, R. Kirkby, E. Frank, and M. Hall. 2002. Multiclass alternating decision trees. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *Proceedings of the Thirteenth European Conference on Machine Learning*, Helsinki, Finland. Berlin: Springer-Verlag, pp. 161–172.

Holte, R. C. 1993. Very simple classification rules perform well on most commonly used datasets. *Machine Learning* 11:63–91.

Huffman, S. B. 1996. Learning information extraction patterns from examples. In S. Wertmer, E. Riloff, and G. Scheler, editors, *Connectionist, statistical, and symbolic approaches to learning for natural language processing*, Springer Verlag, Berlin, pp. 246–260.

Jabbour, K., J. F. V. Riveros, D. Landsbergen, and W. Meyer. 1988. ALFA: Automated load forecasting assistant. *IEEE Transactions on Power Systems* 3(3):908–914.

John, G. H. 1995. Robust decision trees: Removing outliers from databases. In U. M. Fayyad and R. Uthurusamy, editors, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining.* Montreal, Canada. Menlo Park, CA: AAAI Press, pp. 174–179.

———. 1997. *Enhancements to the data mining process.* PhD Dissertation, Computer Science Department, Stanford University, Stanford, CA.

John, G. H., and P. Langley. 1995. Estimating continuous distributions in Bayesian classifiers. In P. Besnard and S. Hanks, editors, *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, Montreal, Canada. San Francisco: Morgan Kaufmann, pp. 338–345.

John, G. H., R. Kohavi, and P. Pfleger. 1994. Irrelevant features and the subset selection problem. In H. Hirsh and W. Cohen, editors, *Proceedings of the Eleventh International Conference on Machine Learning*, New Brunswick, NJ. San Francisco: Morgan Kaufmann, pp. 121–129.

Johns, M. V. 1961. An empirical Bayes approach to nonparametric two-way classification. In H. Solomon, editor, *Studies in item analysis and prediction*. Palo Alto, CA: Stanford University Press.

Kass, R., and L. Wasserman. 1995. A reference Bayesian test for nested hypotheses and its relationship to the Schwarz criterion. *Journal of the American Statistical Association* 90:928–934.

Keerthi, S. S., S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. 2001. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation* 13(3):637–649.

Kerber, R. 1992. Chimerge: Discretization of numeric attributes. In W. Swartout, editor, *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA. Menlo Park, CA: AAAI Press, pp. 123–128.

Kibler, D., and D. W. Aha. 1987. Learning representative exemplars of concepts: An initial case study. In P. Langley, editor, *Proceedings of the Fourth Machine Learning Workshop*, Irvine, CA. San Francisco: Morgan Kaufmann, pp. 24–30.

Kimball, R. 1996. *The data warehouse toolkit*. New York: John Wiley.

Kira, K., and L. Rendell. 1992. A practical approach to feature selection. In D. Sleeman and P. Edwards, editors, *Proceedings of the Ninth International Workshop on Machine Learning*, Aberdeen, Scotland. San Francisco: Morgan Kaufmann, pp. 249–258.

Kittler, J. 1978. Feature set search algorithms. In C. H. Chen, editor, *Pattern recognition and signal processing*. The Netherlands: Sijthoff an Noordhoff.

Koestler, A. 1964. *The act of creation*. London: Hutchinson.

Kohavi, R. 1995a. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Canada. San Francisco: Morgan Kaufmann, pp. 1137–1143.

———. 1995b. The power of decision tables. In N. Lavrac and S. Wrobel, editors, *Proceedings of the Eighth European Conference on Machine Learning*, Iráklion, Crete, Greece. Berlin: Springer-Verlag, pp. 174–189.

———. 1996. Scaling up the accuracy of Naïve Bayes classifiers: A decision tree

hybrid. In E. Simoudis, J. W. Han, and U. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Portland, OR. Menlo Park, CA: AAAI Press, pp. 202–207.

Kohavi, R., and G. H. John. 1997. Wrappers for feature subset selection. *Artificial Intelligence* 97(1–2):273–324.

Kohavi, R., and C. Kunz. 1997. Option decision trees with majority votes. In D. Fisher, editor, *Proceedings of the Fourteenth International Conference on Machine Learning*, Nashville, TN. San Francisco: Morgan Kaufmann, pp. 161–191.

Kohavi, R., and F. Provost, editors. 1998. Machine learning: Special issue on applications of machine learning and the knowledge discovery process. *Machine Learning* 30(2/3): 127–274.

Kohavi, R., and M. Sahami. 1996. Error-based and entropy-based discretization of continuous features. In E. Simoudis, J. W. Han, and U. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Portland, OR. Menlo Park, CA: AAAI Press, pp. 114–119.

Komarek, P., and A. Moore. 2000. A dynamic adaptation of AD trees for efficient machine learning on large data sets. In P. Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning*, Stanford, CA. San Francisco: Morgan Kaufmann, pp. 495–502.

Kononenko, I. 1995. On biases in estimating multivalued attributes. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Canada. San Francisco: Morgan Kaufmann, pp. 1034–1040.

Koppel, M., and J. Schler. 2004. Authorship verification as a one-class classification problem. In R. Greiner and D. Schuurmans, editors, *Proceedings of the Twenty-First International Conference on Machine Learning*, Banff, Alberta, Canada. New York: ACM, pp. 489–495.

Kubat, M., R. C. Holte, and S. Matwin. 1998. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning* 30:195–215.

Kushmerick, N., D. S. Weld, and R. Doorenbos. 1997. Wrapper induction for information extraction. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, Nagoya, Japan. San Francisco: Morgan Kaufmann, pp. 729–735.

Landwehr, N., M. Hall, and E. Frank. 2003. Logistic model trees. In N. Lavrac, D. Gamberger, L. Todorovski, and H. Blockeel, editors, *Proceedings of the Fourteenth European Conference on Machine Learning*, Cavtat-Dubrovnik, Croatia. Berlin: Springer-Verlag. pp. 241–252.

Langley, P. 1996. *Elements of machine learning*. San Francisco: Morgan Kaufmann.

Langley, P., and S. Sage. 1994. Induction of selective Bayesian classifiers. In R. L. de Mantaras and D. Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, Seattle, WA. San Francisco: Morgan

Kaufmann, pp. 399–406.

Langley, P., and H. A. Simon. 1995. Applications of machine learning and rule induction. *Communications of the ACM* 38(11):55–64.

Langley, P., W. Iba, and K. Thompson. 1992. An analysis of Bayesian classifiers. In W. Swartout, editor, *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA. Menlo Park, CA: AAAI Press, pp. 223–228.

Lawson, C. L., and R. J. Hanson. 1995. *Solving least-squares problems.* Philadelphia: SIAM Publications.

le Cessie, S., and J. C. van Houwelingen. 1992. Ridge estimators in logistic regression. *Applied Statistics* 41(1):191–201.

Li, M., and P. M. B. Vitanyi. 1992. Inductive reasoning and Kolmogorov complexity. *Journal Computer and System Sciences* 44:343–384.

Lieberman, H., editor. 2001. *Your wish is my command: Programming by example.* San Francisco: Morgan Kaufmann.

Littlestone, N. 1988. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning* 2(4):285–318.

———. 1989. *Mistake bounds and logarithmic linear-threshold learning algorithms.* PhD Dissertation, University of California, Santa Cruz, CA.

Liu, H., and R. Setiono. 1996. A probabilistic approach to feature selection: A filter solution. In L. Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, Bari, Italy. San Francisco: Morgan Kaufmann, pp. 319–327.

———. 1997. Feature selection via discretization. *IEEE Transactions on Knowledge and Data Engineering* 9(4):642–645.

Mann, T. 1993. *Library research models: A guide to classification, cataloging, and computers.* New York: Oxford University Press.

Marill, T., and D. M. Green. 1963. On the effectiveness of receptors in recognition systems. *IEEE Transactions on Information Theory* 9(11):11–17.

Martin, B. 1995. Instance-based learning: Nearest neighbour with generalisation. MSc Thesis, Department of Computer Science, University of Waikato, New Zealand.

McCallum, A., and K. Nigam. 1998. A comparison of event models for Naïve Bayes text classification. In *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization*, Madison, WI. Menlo Park, CA: AAAI Press, pp. 41–48.

Mehta, M., R. Agrawal, and J. Rissanen. 1996. SLIQ: A fast scalable classifier for data mining. In Apers, P., M. Bouzeghoub, and G. Gardarin, *Proceedings of the Fifth International Conference on Extending Database Technology*, Avignon, France. New York: Springer-Verlag.

Melville, P., and R. J. Mooney. 2005. Creating diversity in ensembles using artificial data. *Information Fusion* 6(1):99–111.

Michalski, R. S., and R. L. Chilausky. 1980. Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *International Journal of Policy Analysis and Information Systems* 4(2).

Michie, D. 1989. Problems of computer-aided concept formation. In J. R. Quinlan, editor, *Applications of expert systems, Vol. 2.* Wokingham, England: Addison-Wesley, pp. 310–333.

Minsky, M., and S. Papert. 1969. *Perceptrons.* Cambridge, MA: MIT Press.

Mitchell, T. M. 1997. *Machine learning.* New York: McGraw Hill.

Mitchell, T. M., R. Caruana, D. Freitag, J. McDermott, and D. Zabowski. 1994. Experience with a learning personal assistant. *Communications of the ACM* 37(7):81–91.

Moore, A. W. 1991. *Efficient memory-based learning for robot control.* PhD Dissertation, Computer Laboratory, University of Cambridge, UK.

———. 2000. The anchors hierarchy: Using the triangle inequality to survive high-dimensional data. In C. Boutilier and M. Goldszmidt, editors, *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, Stanford, CA. San Francisco: Morgan Kaufmann, pp. 397–405.

Moore, A. W., and M. S. Lee. 1994. Efficient algorithms for minimizing cross validation error. In W. W. Cohen and H. Hirsh, editors, *Proceedings of the Eleventh International Conference on Machine Learning*, New Brunswick, NJ. San Francisco: Morgan Kaufmann, pp. 190–198.

———. 1998. Cached sufficient statistics for efficient machine learning with large datasets. *Journal Artificial Intelligence Research* 8:67–91.

Moore, A. W., and D. Pelleg. 2000. *X*-means: Extending *k*-means with efficient estimation of the number of clusters. In P. Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning*, Stanford, CA. San Francisco: Morgan Kaufmann, pp. 727–734.

Nadeau, C., and Y. Bengio. 2003. Inference for the generalization error. *Machine Learning* 52(3):239–281.

Nahm, U. Y., and R. J. Mooney. 2000. Using information extraction to aid the discovery of prediction rules from texts. *Proceedings of the Workshop on Text Mining at the Sixth International Conference on Knowledge Discovery and Data Mining*, Boston, MA, pp. 51–58.

Nie, N. H., C. H. Hull, J. G. Jenkins, K. Steinbrenner, and D. H. Bent. 1970. *Statistical package for the social sciences.* New York: McGraw Hill.

Nigam, K., and R. Ghani. 2000. Analyzing the effectiveness and applicability of co-

training. *Proceedings of the Ninth International Conference on Information and Knowledge Management*, McLean, VA. New York: ACM, pp. 86–93.

Nigam, K., A. K. McCallum, S. Thrun, and T. M. Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning* 39(2/3):103–134.

Nilsson, N. J. 1965. *Learning machines.* New York: McGraw Hill.

Omohundro, S. M. 1987. Efficient algorithms with neural network behavior. *Journal of Complex Systems* 1(2):273–347.

Paynter, G. W. 2000. *Automating iterative tasks with programming by demonstration.* PhD Dissertation, Department of Computer Science, University of Waikato, New Zealand.

Piatetsky-Shapiro, G., and W. J. Frawley, editors. 1991. *Knowledge discovery in databases.* Menlo Park, CA: AAAI Press/MIT Press.

Platt, J. 1998. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in kernel methods: Support vector learning.* Cambridge, MA: MIT Press.

Provost, F., and T. Fawcett. 1997. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, Huntington Beach, CA. Menlo Park, CA: AAAI Press.

Pyle, D. 1999. *Data preparation for data mining.* San Francisco: Morgan Kaufmann.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning* 1(1):81–106.

———. 1992. Learning with continuous classes. In N. Adams and L. Sterling, editors, *Proceedings of the Fifth Australian Joint Conference on Artificial Intelligence*, Hobart, Tasmania. Singapore: World Scientific, pp. 343–348.

———. 1993. *C4.5: Programs for machine learning.* San Francisco: Morgan Kaufmann.

Rennie, J. D. M., L. Shih, J. Teevan, and D. R. Karger. 2003. Tackling the poor assumptions of Naïve Bayes text classifiers. In T. Fawcett and N. Mishra, editors, *Proceedings of the Twentieth International Conference on Machine Learning*, Washington, DC. Menlo Park, CA: AAAI Press, pp. 616–623.

Ricci, F., and D. W. Aha. 1998. Error-correcting output codes for local learners. In C. Nedellec and C. Rouveird, editors, *Proceedings of the European Conference on Machine Learning*, Chemnitz, Germany. Berlin: Springer-Verlag, pp. 280–291.

Richards, D., and P. Compton. 1998. Taking up the situated cognition challenge with ripple-down rules. *International Journal of Human-Computer Studies* 49(6):895–926.

Ripley, B. D. 1996. *Pattern recognition and neural networks.* Cambridge, UK: Cambridge University Press.

Rissanen, J. 1985. The minimum description length principle. In S. Kotz and N. L.

Johnson, editors, *Encyclopedia of Statistical Sciences, Vol. 5.* New York: John Wiley, pp. 523–527.

Rousseeuw, P. J., and A. M. Leroy. 1987. *Robust regression and outlier detection.* New York: John Wiley.

Sahami, M., S. Dumais, D. Heckerman, and E. Horvitz. 1998. A Bayesian approach to filtering junk email. In *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization*, Madison, WI. Menlo Park, CA: AAAI Press, pp. 55–62.

Saitta, L., and F. Neri. 1998. Learning in the "real world." *Machine Learning* 30(2/3):133–163.

Salzberg, S. 1991. A nearest hyperrectangle learning method. *Machine Learning* 6(3):251–276.

Schapire, R. E., Y. Freund, P. Bartlett, and W. S. Lee. 1997. Boosting the margin: A new explanation for the effectiveness of voting methods. In D. H. Fisher, editor, *Proceedings of the Fourteenth International Conference on Machine Learning*, Nashville, TN. San Francisco: Morgan Kaufmann, pp. 322–330.

Scheffer, T. 2001. Finding association rules that trade support optimally against confidence. In L. de Raedt and A. Siebes, editors, *Proceedings of the Fifth European Conference on Principles of Data Mining and Knowledge Discovery*, Freiburg, Germany. Berlin: Springer-Verlag, pp. 424–435.

Schölkopf, B., and A. J. Smola. 2002. *Learning with kernels: Support vector machines, regularization, optimization, and beyond.* Cambridge, MA: MIT Press.

Schölkopf, B., P. Bartlett, A. J. Smola, and R. Williamson. 1999. Shrinking the tube: A new support vector regression algorithm. *Advances in Neural Information Processing Systems, Vol. 11.* Cambridge, MA: MIT Press, pp. 330–336.

Sebastiani, F. 2002. Machine learning in automated text categorization. *ACM Computing Surveys* 34(1):1–47.

Seeger, M. 2001. Learning with labeled and unlabeled data. Technical Report, Institute for Adaptive and Neural Computation, University of Edinburgh, UK.

Seewald, A. K. 2002. How to make stacking better and faster while also taking care of an unknown weakness. *Proceedings of the Nineteenth International Conference on Machine Learning*, Sydney, Australia. San Francisco: Morgan Kaufmann, pp. 54–561.

Seewald, A. K., and J. Fürnkranz. 2001. An evaluation of grading classifiers. In F. Hoffmann, D. J. Hand, N. M. Adams, D. H. Fisher, and G. Guimarães, editors, *Proceedings of the Fourth International Conference on Advances in Intelligent Data Analysis*, Cascais, Portugal. Berlin: Springer-Verlag, pp.115–124.

Shafer, R., R. Agrawal, and M. Metha. 1996. SPRINT: A scalable parallel classifier for data mining. In T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, editors, *Proceedings of the Second International Conference on Very Large Databases*, Mumbai (Bombay), India. San Francisco: Morgan Kaufmann, pp. 544–555.

Shawe-Taylor, J., and N Cristianini. 2004. *Kernel methods for pattern analysis.* Cambridge, UK: Cambridge University Press.

Smola, A. J., and B. Schölkopf. 2004. A tutorial on support vector regression. *Statistics and Computing* 14(3):199–222.

Soderland, S., D. Fisher, J. Aseltine, and W. Lehnert. 1995. Crystal: inducing a conceptual dictionary. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Canada. Menlo Park, CA: AAAI Press, pp. 1314–1319.

Stevens, S. S. 1946. On the theory of scales of measurement. *Science* 103:677–680.

Stone, P., and M. Veloso. 2000. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots* 8(3):345–383.

Swets, J. 1988. Measuring the accuracy of diagnostic systems. *Science* 240: 1285–1293.

Ting, K. M. 2002. An instance-weighting method to induce cost-sensitive trees. *IEEE Transactions on Knowledge and Data Engineering* 14(3):659–665.

Ting, K. M., and I. H. Witten. 1997a. Stacked generalization: When does it work? In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, Nagoya, Japan. San Francisco: Morgan Kaufmann, pp. 866–871.

———. 1997b. Stacking bagged and dagged models. In D. H. Fisher, editor, *Proceedings of the Fourteenth International Conference on Machine Learning*, Nashville, TN. San Francisco: Morgan Kaufmann, pp. 367–375.

Turney, P. D. 1999. Learning to extract key phrases from text. Technical Report ERB-1057, Institute for Information Technology, National Research Council of Canada, Ottawa, Canada.

U. S. House of Representatives Subcommittee on Aviation. 2002. Hearing on aviation security with a focus on passenger profiling, February 27, 2002. [http://www.house.gov/transportation/aviation/02-27-02/02-27-02memo.html].

Vafaie, H., and K. DeJong. 1992. Genetic algorithms as a tool for feature selection in machine learning. In *Proceedings of the International Conference on Tools with Artificial Intelligence.* Arlington, VA: IEEE Computer Society Press, pp. 200–203.

van Rijsbergen, C. A. 1979. *Information retrieval.* London: Butterworths.

Vapnik, V. 1999. *The nature of statistical learning theory*, second edition. New York: Springer-Verlag.

Wang, Y., and I. H. Witten. 1997. Induction of model trees for predicting continuous classes. In M. van Someren and G. Widmer, editors, *Proceedings of the Poster Papers of the European Conference on Machine Learning.* Prague: University of Economics, Faculty of Informatics and Statistics, pp. 128–137.

———. 2002. Modeling for optimal probability prediction. In C. Sammut and A.

Hoffmann, editors, *Proceedings of the Nineteenth International Conference on Machine Learning*, Sydney, Australia. San Francisco: Morgan Kaufmann, pp. 650–657.

Webb, G. I. 2000. MultiBoosting: A technique for combining boosting and wagging. *Machine Learning* 40(2):159–196.

Webb, G. I., J. Boughton, and Z. Wang. 2005. Not so Naïve Bayes: Aggregating one-dependence estimators. *Machine Learning* 58(1):5–24.

Weiser, M. 1996. Open house. *Review*, the Web magazine of the Interactive Telecommunications Program of New York University. March.

Weiser, M., and J. S. Brown. 1997. The coming age of calm technology. In P. J. Denning and R. M. Metcalfe, editors, *Beyond calculation: The next fifty years.* New York: Copernicus, pp. 75–86.

Weiss, S. M., and N. Indurkhya. 1998. *Predictive data mining: A practical guide.* San Francisco: Morgan Kaufmann.

Wettschereck, D., and T. G. Dietterich. 1995. An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. *Machine Learning* 19(1):5–28.

Wild, C. J., and G. A. F. Seber. 1995. *Introduction to probability and statistics.* Department of Statistics, University of Auckland, New Zealand.

Winston, P. H. 1992. *Artificial intelligence.* Reading, MA: Addison-Wesley.

Witten, I. H. 2004. Text mining. In M. P. Singh, editor, *Practical handbook of internet computing.* Boca Raton, FL: CRC Press.

Witten, I. H., Z. Bray, M. Mahoui, and W. Teahan. 1999a. Text mining: A new frontier for lossless compression. In J. A. Storer and M. Cohn, editors, *Proceedings of the Data Compression Conference*, Snowbird, UT. Los Alamitos, CA: IEEE Computer Society Press, pp. 198–207.

Witten, I. H., A. Moffat, and T. C. Bell. 1999b. *Managing gigabytes: Compressing and indexing documents and images*, second edition. San Francisco: Morgan Kaufmann.

Wolpert, D. H. 1992. Stacked generalization. *Neural Networks* 5:241–259.

Yang, Y., and G. I. Webb. 2001. Proportional *k*-interval discretization for Naïve Bayes classifiers. In L. de Raedt and P. Flach, editors, *Proceedings of the Twelfth European Conference on Machine Learning*, Freiburg, Germany. Berlin: Springer-Verlag, pp. 564–575.

Yurcik, W., J. Barlow, Y. Zhou, H. Raje, Y. Li, X. Yin, M. Haberman, D. Cai, and D. Searsmith. 2003. Scalable data management alternatives to support data mining heterogeneous logs for computer network security. In *Proceedings of the Workshop on Data Mining for Counter Terrorism and Security*, San Francisco. Society for International and Applied Mathematics, Philadelphia, PA.

Zheng, Z., and G. Webb. 2000. Lazy learning of Bayesian rules. *Machine Learning* 41(1):53–84.

索　引

索引中的页码为英文原书页码，与书中边栏页码一致。

A

activation function（激活函数），234
acuity（灵敏度），258
AdaBoost，328
AdaBoost.M1，321,416
Add，395
AddCluster，396，397
AddExpression，397
additive logistic regression（叠加logistic回归），327-328
additive regression（叠加回归），325-327
AdditiveRegression，416
AddNoise，400
AD（all-dimensions）tree（AD树），280-283
ADTree，408
advanced methods（高级方法）. *See* implementation—real-world schemes
adversarial data mining（对抗性的数据挖掘），356-358
aggregation（整合），appropriate degree in data warehousing（数据仓库的适当（整合）度），53
Akaike Information Criterion（AIC）（Akaike信息标准），277
Alberta Ingenuity Centre for Machine learning（阿尔贝塔机器学习研发中心），38
algorithms（算法）
additive logistic regression（叠加logistic回归），327
advanced methods（高级方法），187-283. *See also* implementation——real-world schemes
association rule mining（关联规则挖掘），112-119
bagging（装袋），319
basic methods（基本方法），83-142. *See also* algorithms-basic methods
Bayesian network learning（贝叶斯网络学习），277-283
clustering（聚类），136-139
clustering in Weka（在Weka中的聚类），418-419
covering（覆盖），105-112
decision tree induction（决策树归纳），97-105
divide-and-conquer（分治法），107
EM，265-266
expanding examples into partial tree（将样本展开成局部树），208
filtering in Weka（Weka中的过滤），393-403. *See also* filtering algorithms
incremental（递增的），346
instance-based learning（基于实例的学习），128-136
learning in Weka（在Weka中学习），403-414. *See also* learning algorithms
linear models（线性模型），119-128
metalearning in Weka（在Weka中的元学习），414-418
1R method（1R法），84-88
perceptron learning rule（感知器学习规则），124
RIPPER rule learner（RIPPER 规则学习器），206
rule formation－incremental reduced－error pruning（规则形成——递增减少误差修剪法），205
separate-and-conquer（割治法），112
statistical modeling（统计建模），88-97
stochastic（随机化），348
Winnow（Winnow算法），127
See also individual subject headings.
all-dimensions（AD）tree（AD树），280-283
alternating decision tree（交互式决策树），329,330，343
Analyze panel（分析面板），443-445
analyzing purchasing patterns（分析购买模式），27
ancestor-of（先辈），48
anomalies（异情），314-315
anomaly detection systems（异情侦察系统），357
antecedent（前提），of rule（规则），65
AODE，405
Apriori，419
Apriori method（先验方法），141
area under the curve（AUC）（曲线下方区域），173
ARFF format（ARFF 格式），53-55

converting files to（将文件转换为）、380-382
Weka、370、371
ARFFLoader、382、427
arithmetic underflow（算法下溢）、276
assembling the data（数据集中）、52-53
assessing performance of learning scheme（学习方案性能评估）、286
assignment of key phrases（赋予关键词组）、353
Associate panel、392
association learning（关联学习）、43
association rules（关联规则）、69-70、112-119
binary attributes（二值属性）、119
generating rules efficiently（有效地产生规则）、117-118
item sets（项集）、113、114-115
Weka、419-420
association-rule learners in Weka（Weka 中的关联规则学习器）、419-420
attackers（攻击者）、357
Attribute、451
attribute（）、480
attribute discretization（属性离散）See discretizing numeric attributes
attribute-efficient（有效-属性）、128
attribute evaluation methods in Weka（Weka中的属性评估方法）、421、422-423
attribute filters in Weka（Weka 中的属性过滤器）、394,395-400,402-403
attributeIndices、382
attribute noise（属性干扰）、313
attribute-relation file format（属性-关系文件格式）. *See* ARFF format attributes（属性）、49-52
adding irrelevant（增加无关的）、288
Boolean（布尔）、51
class（类）、53
as columns in tables（如表中的列）、49
combinations of（组合）、65
continuous（连续的）、49
discrete（离散的）、51
enumerated（枚举的）、51
highly branching（高度分支的）、86
identification code（标识码）、86
independent（独立的）、267
integer-valued（整数值的）、49
nominal（名词性的）、49
non-numeric（非数值型的）、17
numeric（数值型的）、49
ordinal（序数型的）、51
relevant（相关的）、289
rogue（不良的）、59
selecting（选择）、288
subsets of values in（值的子集）、80
types in ARFF format（ARFF格式种类）、56
weighting（加权）、237
See also orderings
AttributeSelectedClassifier、417
attribute selection（属性选择）、286-287、288-296
attribute evaluation methods in Weka（在Weka中的属性评估方法）、421、422-423
backward elimination（反向消除）、292、294
beam search（限定范围搜索）、293
best-first search（最佳优先搜索）、293
forward selection（正向选择）、292、294
race search（竞赛搜索）、295
schemata search（模式搜索）、295
scheme-independent selection（独立于方案的选择）、290-292
scheme-specific selection（特定方案选择）、294-296
searching the attribute space（搜索属性空间）、292-294
search methods in Weka（在Weka中的搜索方法）、421、423-425
Weka、392-393、420-425
AttributeSelection、403
attribute subset evaluators in Weka（在Weka 中的属性子集评估器）、421、422
AttributeSummarizer、431
attribute transformations（属性转换）、287、305-311
principal components analysis（主分量分析）、306-309
random projections（随机投影）、309
text to attribute vectors（从文本到属性向量）、309-311
time series（时间序列）、311
attribute weighting method（属性加权方法）、237-238
AUC（area under the curve）（曲线下方区域）、173
audit logs（审核日志）、357
authorship ascription（作者归属）、353
AutoClass、269-270、271
automatic data cleansing（自动数据清理）、287、312-315
anomalies（异情）、314-315

improving decision trees（改善决策树），312-313
robust regression（稳健回归），313-314
automatic filtering（自动过滤），315
averaging over subnetworks（平均子网络），283
axis-parallel class boundaries（轴平行类边界），242

B

background knowledge（背景知识），348
backpropagation（反向传播），227-233
backtracking（返回），209
backward elimination（反向消除），292，294
backward pruning（反向修剪），34，192
bagging（装袋），316-319
Bagging，414-415
bagging with costs（考虑成本的装袋），319-320
bag of words（词语袋），95
balanced Winnow（平衡的Winnow算法），128
ball tree（球树），133-135
basic methods（基本方法）*See* algorithms – basic methods
batch learning（批处理学习），232
Bayes，Thomas（贝叶斯，托马斯），141
Bayesian classifier（贝叶斯分类器）*See* Naïve Bayes
Bayesian clustering（贝叶斯聚类），268-270
Bayesian multinet（贝叶斯复网），279-280
Bayesian network（贝叶斯网络），141，271-283
AD tree（AD树），280-283
Bayesian multinet（贝叶斯复网），279-280
caveats（告诫），276，277
counting（计数），280
K2，278
learning（学习），276-283
making predictions（做预测），272-276
Markov blanket（马尔可夫毯），278-279
multiplication（乘法），275
Naïve Bayes classifiers（朴素贝叶斯分类器），278
network scoring（网络评分），277
simplifying assumptions（简化假设），272
structure learning by conditional independence tests（条件独立测试的结构学习），280
TAN（Tree Augmented Naïve Bayes）（树扩展型朴素贝叶斯），279
Weka，403-406
Bayesian network learning algorithms（贝叶斯网络学习算法），277-283
Bayesian option trees（贝叶斯选择树），328-331,343
Bayesians（贝叶斯），141
Bayesian scoring metrics（贝叶斯评分量度），277-280，283
Bayes information（贝叶斯信息），271
BayesNet，405
Bayes's rule（贝叶斯规则），90，181
beam search（限定范围搜索），34，293
beam width（束宽），34
beer purchases（啤酒购买），27
Ben Ish Chai，358
Bernoulli process（伯努利程式），147
BestFirst，423
best-first search（最佳优先搜索），293
best-matching node（最合适的节点），257
bias（偏差），32
defined（定义），318
language（语言），32-33
multilayer perceptrons（多层感知器），225，226
overfitting-avoidance（避免过度拟合），34-35
perceptron learning rule（感知器学习规则），124
search（搜索），33-34
what is it（它是什么），317
bias-variance decomposition（偏差-方差分解），317，318
big data（massive datasets）（大型数据），346-349
binning（装箱）
equal-frequency（等频率），298
equal-interval（等区间），298
equal-width（等宽），342
binomial coefficient（二项系数），218
bits（位），102
boolean（布尔），51，68
boosting（提升），321-325，347
boosting in Weka（Weka中的提升），416
bootstrap aggregating（自引导整合），318
bootstrap estimation（自引导估计），152-153
British Petroleum（英国石油公司），28
buildClassifier()，453，472，482

C

C4.5，105，198-199
C5.0，199
calm computing（平静的计算），359,362
capitalization conventions（大写习惯），310
CAPPS（Computer Assisted Passenger PreScreening System）（计算机辅助旅客预筛选系统），357
CART（Classification And Regression Tree）（分类及回

归树)、29、38,199、253
categorical attributes （范畴属性)、49. *See also* nominal attributes
category utility （类别效用)、260-262
casual relations (因果关系)、350
CfsSubsetEval、422
chain rule (probability theory) (链规则 (概率理论))、275
character sets (字符集)、310
ChiSquaredAttributeEval、423
chi-squared test (χ^2测试)、302
circular ordering (循环排序)、51
city-block metric (city-block度量法)、129
ClassAssigner、431
class attribute （类属性)、43
class distribution (类分布)、304
class hierarchy in Weka （Weka中的类层次结构)、471-483
classification (分类)、121
Classification And Regression Tree （CART) (分类及回归树) ,29、38、199、253
classification learning (分类学习)、43
classification problems (分类问题)、42
classification rules (分类规则)、65-69、200-214
  converting decision trees to (将决策树转换为)、198
  criteria for choosing tests （测试选择的标准)、200-201
  decision list （决策列)、11
  different from association rules (与关联规则不同)、42
  global optimization （全局优化)、205-207
  good （worthwhile） rules (好的 (有价值的) 规则)、202-205
  missing values （残缺值)、201-202
  numeric attributes （数值属性)、202
  partial decision tree (局部决策树)、207-210
  pruning （修剪)、203、205
  replicated subtree problem （重复子树问题)、66-68
  RIPPER rule learner （RIPPER 规则学习器)、205
  rules with exceptions （含例外的规则)、210-213
  Weka、408-409
Classifier、453
classifier in Weka （Weka 中的分类器)、366、471-483
classifier algorithms （分类器算法) .*See* learning algorithms
classifier errors in Weka (Weka中的分类器误差)、379
ClassifierPerformanceEvaluator、431
classifiers package (分类器包)、453-455
ClassifierSubsetEval、422
Classifier superclass、480
classifyInstance ()、453、480-481
Classify panel、373、384
classify text files into two categories （将文本文件分成两个类别)、461-469
Classit、271
class noise （类别干扰)、313
ClassOrder、403
class summary、451
ClassValuePicker、431
cleaning data （清理数据)、52、60
  automatic (自动的)、312
closed world assumption (闭合世界的假设)、45
ClustererPerformanceEvaluator、431
clustering （聚类)、43、136-139、254-271
  anomalies (异情)、258
  basic assumption (基本假设)、270
  Bayesian (贝叶斯)、268-270
  category utility (类别效用)、260-262
  Cluster panel （Weka)、391-392
  document (文件)、353
  EM algorithm (EM 算法)、265-266
  faster distance calculations (快速距离计算)、138-139
  hierarchical (等级的)、139
  how many clusters (多少聚类)、254-255
  incremental (递增的)、255-260
  k-means (*k*均值)、137-138
  MDL principal (MDL原理)、183-184
  merging (合并)、257
  mixture model (混合模型)、262-264、266-268
  output (输出)、81-82
  probability-based (基于概率的)、262-265
  RBF network （RBF网络)、234
  splitting (分裂)、254-255、257
  unlabeled data （无标签数据)、337-339
clustering algorithms in Weka （Weka中的聚类算法)、418-419
clustering for classification （用于分类的聚类)、337
ClusterMembership、396、397
Cluster panel、391-392
Cobweb、271
Cobweb in Weka、419

co-EM，340
column separation（列分隔），336
combining multiple models（组合多个模型），287，315-336
  additive logistic regression（叠加logistic 回归），327-328
  additive regression（叠加回归），325-327
  bagging（装袋），316-319
  bagging with costs（考虑成本的装袋），319-320
  boosting（提升），321-325
  error-correcting output codes（误差纠正输出编码），334-336
  logistic model trees（logistic 模型树），331
  option trees（选择树），328-331
  randomization（随机化），320-321
  stacking（堆栈），332-334
command-line interface（命令行界面），449-459
  class（类），450，452
  classifiers package（分类器包），453-455
  core package（核心包），451,452
  generic options（一般选项），456-458
  instance（实例），450
  Javadoc indices（Java文本索引），456
  package（包），451,452
  scheme-specific options（仅适用于某个具体模式的选项），458-459
  starting up（开始着手），449-450
  weak.associations，455
  weak.attributeSelection，455
  weak.clusterers，455
  weak.estimators，455
  weak.filters，455
command-line options（命令行选项），456-459
comma-sperated value（CSV）format（以逗号来分隔值的格式），370，371
comparing data mining methods（比较数据挖掘方法），153-157
ComplementNaiveBayes，405
compression techniques（压缩技术），362
computational learning theory（计算学习理论），324
computeEntropy（），480
Computer Assisted Passenger Pre-Screening System（CAPPS），（计算机辅助旅客预筛选系统），357
computer network security（计算机网络安全），357
computer software（计算机软件）. *See* Weka workbench
concept（概念），42
concept description（概念描述），42
concept description language（概念描述语言），32
concept representation（概念表达），82. *See also* knowledge representation
conditional independence（条件独立），275
conditional likelihood for scoring networks（网络评分的条件似然率），280，283
confidence（置信度），69，113,324
Confidence，420
confidence tests（置信度测试），154-157，184
conflict resolution strategies（冲突解决策略），82
confusion matrix（混淆矩阵），163
conjunction（联合），65
ConjunctiveRule，408-409
consensus filter（共识过滤器），342
consequent，of rule（由规则所致的），65
ConsistencySubsetEval，422
constrained quadratic optimization（受限的二次优化），217
consumer music（音乐制品），359
contact lens data（隐形眼镜数据），6，13-15
continuous attributes（连续属性），49. *See also* numeric attributes
continuous monitoring（持续监视），28-29
converting discrete to numeric attributes（将离散转换为数值属性），304-305
convex hull（凸包），171，216
Conviction，420
Copy，395
core package（核心包），451,452
corrected resampled t-test（纠正重复取样t测试），157
correlation coefficient（关联系数），177-179
cost curves（成本曲线），173-176
cost matrix（成本矩阵），164-165
cost of errors（误差成本），161-176
  bagging（装袋），319-320
  cost curves（成本曲线），173-176
  cost-sensitive classification（成本敏感分类），164-165
  cost-sensitive learning（成本敏感学习），165-166
  Kappa statistic（Kappa统计量），163-164
  lift charts（上升图），166-168
  recall-precision curves（反馈率-精确率曲线），171-172
  ROC curves（ROC曲线），168-171
cost-sensitive classification（成本敏感分类），164-165

CostSensitiveClassifier, 417
cost-sensitive learning（成本敏感学习）, 165-166
cost-sensitive learning in Weka（在Weka中的成本敏感学习）, 417
co-training （联合训练）, 339-340
covariance matrix （协方差矩阵）, 267, 307
coverage, of association rules（覆盖量，关联规则的）, 69
covering algorithm （覆盖算法）, 106-111
cow culling （母牛选择）, 3-4, 37, 161-162
CPU performance data （CPU性能数据）, 16-17
credit approval （信用批准）, 22-23
cross-validated ROC curves （经交叉验证的ROC曲线）, 170
cross-validation（交叉验证）, 149-152, 326
  inner （内部的）, 286
  outer（外部的）, 286
  repeated （重复的）,144
CrossValidationFoldMaker, 428, 431
CSV format （CSV 格式）, 370, 371
CSVLoader, 381
cumulative margin distribution in Weka（Weka 中的累积边差分布）, 458
curves （曲线）
  cost（成本）, 173
  lift（上升）, 166
  recall-precision（反馈率-精确率）, 171
  ROC, 168
customer support and service （客户支持和服务）, 28
cutoff parameter （截止参数）, 260
CVParameterSelection, 417
cybersecurity（计算机网络安全）, 29

D

dairy farmers（畜牧业农场主） (New Zealand), 3-4, 37, 161-162
data assembly（数据装配）, 52-53
data cleaning（数据清理）, 52-60. *See also* automatic data cleansing
data engineering（数据处理）.*See* engineering input and output
data integration（数据集成）, 52
data mining（数据挖掘）, 4-5, 9
data ownership rights（数据拥有权）, 35
data preparation（数据准备）, 52-60
data transformation（数据转换） *See* attribute transformations
DataVisualizer, 389, 390, 430
data warehouse（数据仓库）, 52-53
data attributes（数据属性）, 55
decision list（决策列）, 11,67
decision nodes（决策节点）, 328
decision stump（决策树桩）, 325
DecisionStump, 407, 453, 454
decision table（决策表）, 62, 295
DecisionTable, 408
decision tree（决策树）, 14, 62-65, 97-105
  complexity of induction（归纳复杂度）, 196
  converting to rules（转换成规则）, 198
  data cleaning（数据清理）, 312-313
  error rates（误差率）, 192-196
  highly branching attributes（高度分支的属性）, 102-105
  missing values（残缺值）, 63, 191-192
  multiclass case（多类情形）, 107
  multivariate（多元的）, 199
  nominal attribute（名词性属性）, 62
  numeric attribute（数值属性）, 62, 189-191
  partial（部分的）, 207-210
  pruning （修剪）, 192-193, 312
  replicated subtree（重复子树）, 66
  rules（规则）, 198
  subtree raising（子树上升）, 193, 197
  subtree replacement（子树置换）, 192-193, 197
  three-way split（三叉分裂）, 63
  top-down induction（自上而下的归纳）, 97-105, 196-198
  two-way split（二叉分裂）, 62
  univariate（单变量的）, 199
  Weka, 406-408
  Weka's User Classifier facility（Weka用户分类工具）, 63-65
Decorate, 416
deduction（推导）, 350
default rule（缺省规则）, 110
degrees of freedom（自由度）, 93, 155
delta, 311
dendrograms （树状图）, 82
denormalization （反向正常化）, 47
density function（密度函数）, 93
diagnosis（诊断）, 25-26
dichotomy （二分法）, 51
directed acyclic graph（有向无环图）, 272

direct marketing（直销市场），27
discrete attributes（离散属性），50. *See also* nominal attributes
Discretize，396，398，402
discretizing numeric attributes（离散数值属性），287，296-305
chi-squared test（卡方测试），302
converting discrete to numetic attributes（将离散属性转换为数值属性），304-305
entropy-based discretization（基于熵的离散），298-302
error-based discretization（基于误差的离散），302-304
global discretization（全局离散），297
local discretization（局部离散），297
supervised discretization（有指导离散），297-298
unsupervised discretization（无指导离散），297-298
Weka，398
disjunction（或），32，65
disjunctive normal form（析取范式），69
distance functions（距离函数），128-129，239-242
distributed experiments in Weka（Weka中的分布式实验），445
distribution（分布），304
distributionForInstance（），453，481
divide-and-conquer（分治）*See* decision tree
document classification（文件分类），94-96，352-353
document clustering（文件聚类），353
domain knowledge（领域知识），20，33，349-351
double-consequent rules（双重结果规则），118
duplicate data（重复数据），59
dynamic programming（动态编程），302

E

early stopping（提前停止），233
easy instances（易处理实例），322
ecological applications（生态学领域的应用程序），23,38
eigenvalue（特征值），307
eigenvector（特征向量），307
Einstein，Albert（爱因斯坦·艾伯特），180
electricity supply（电力供应），24-25
electromechanical diagnosis application（机电诊断应用），144
11-point averge recall（11点平均反馈率），172
EM，418
EM algorithm（EM算法），265-266
EM and co-training（EM和联合训练），340-341
EM procedure（EM程序），337-338
embeded machine learning（嵌入式机器学习），461-469
engineering input and output（处理输入和输出），285-343
attribute selection（属性选择），288-296
combining multiple models（组合多种模型），315-336
data cleansing（数据清理），312-315
discretizing numeric attributes（离散数值属性），296-305
unlabeled data（无标签数据），337-341
See also individual subject headings
entity extraction（实体提炼），353
entropy（熵），102
entropy-based discretization（基于熵的离散），298-302
enumerated attributes（枚举的属性），50. *see also* nominal attributes
enumerating the concept space（枚举概念空间），31-32
Epicurus（伊壁鸠鲁），183
epoch（纪元），412
equal-frequency binning（等频率装箱），298
equal-interval binning（等区间装箱），298
equal-width binning（等宽装箱），342
erroneous values（错误的值），59
error-based discretization（基于误差的离散），302-304
error-correcting output codes（误差纠正输出编码），334-336
error log（错误日志），378
error rate（误差率）
bias（偏差），317
cost of errors（误差成本），*See* cost of errors
decision tree（决策树），192-196
defined（定义的），144
training data（训练数据），145
"Essay towards solving a problem in the doctrine of chances，An"（一篇倾向于运用机会学说解决问题的论文）（Bayes），141
ethics（道德规范），35-37
Euclidean distance（欧几里得距离），78，128，129，237
evaluation（评估），143-185
bootstrap procedure（自引导程序），152-153
comparing data mining methods（比较数据挖掘方法）153-157
cost of errors（误差成本），161-176. *See also* cost of errors

cross-validation（交叉验证），149-152
leave-one-out cross-validation（留一交叉验证 ）151-152
MDL principal（MDL原理），179-184
numeric prediction（数值预测），176-179
predicting performance（预测性能），146-149
predicting probabilities（预测概率），157-161
training and testing（训练和测试），144-146
evaluation（），482
evaluation components in Weka（Weka中的评估组件），430，431
Evaluation panel，431
example problems（样例问题）
contact lens data（隐形眼镜数据），6，13-15
CPU performance data（CPU性能数据），16-17
iris dataset（鸢尾花数据集），15-16
labor negotiations data（劳资协商数据），17-18，19
soybean data（大豆数据），18-22
weather problem（天气问题），10-12
exceptions（例外），70-73，210-213
exclusive-or problem（异或问题），67
exemplar（样本）
defined（定义的），236
generalized（被推广的），238-239
noisy（干扰的），236-237
redundant（重复的），236
exemplar generalization（样本推广），238-239，243
ExhaustiveSearch，424
Expand all paths，408
expectation（期望），265，267
expected error（期望误差），174
expected success rate（期望成功率），147
Experimenter（实验者），437-447
advanced panel（高级面板），443-445
Analyze panel（分析面板），443-445
analyzing the results（分析结果），440-441
distributing processing over several machines（将处理过程分布于多个机器），445-447
running an experiment（运行一个实验），439-440
simple setup（简单配置），441-442
starting up（开始），438-441
subexperiments（子实验），447
Explorer（探索者），369-425
ARFF，370，371，380-382
Associate panel，392
association-rule learners（关联规则学习器），419-420
attribute evaluation methods（属性评估方法），421，422-423
attribute selection（属性选择），392-393，420-425
boosting（提升），416
classifier errors（分类器误差），379
Classify panel，384
clustering algorithms（聚类算法），418-419
Cluster panel，391-392
CSV format（CSV格式），370，371
error log（错误日志），378
file format converters（文件格式转换器），380-382
filtering algorithms（过滤算法），393-403
filter（过滤器），382-384
J4.8，373-377
learning algorithms，（学习算法），403-414. *See also* learning algorithms
metalearning algorithms（元学习算法），414-418
models（模型），377-378
panels（面板），380
Preprocess panel，380
search methods（搜索方法），421，423-425
Select attributes panel，392-393
starting up（开始），369-379
supervised filters（有指导过滤器），401-403
training / testing learning schemes（训练/测试学习方案）384-387
unsupervised attribute filters（无指导属性过滤器）395-400
unsupervised instance filters（无指导实例过滤器）400-401
User Classifier（用户自己的分类器），388-391
Visualize panel，393
extraction problems（提炼问题），353，354

F

Fahrenheit（华氏温度），Daniel，51
fallback heuristic（后退启发式的），239
false negative（FN）（错误的否定），162
false positive（FP）（错误的肯定），162
false positive rate（错误肯定率），163
False positive rate，378
Familiar，360
family tree（家族树），45
tabular representation of（表格式排列），46
FarthesFirst，419
features（属性），*See* attributes
feature selection（属性选择），341. *See also* attribute selection

feedforward networks（前馈网络），233
fielded applications（领域应用），22
continuous monitoring（持续监视），28-29
customer support and service（客户支持和服务），28
cybersecurity（网络安全），29
diagnosis（诊断），25-26
ecological applications（生态领域的应用），23，28
electricity supply（电力供应），24-25
hazard detection system（危险探测系统），23-24
load forecasting（负载预测），24-25
loan application（借贷应用），22-23
manufacturing processes（制造工艺），28
marketing and sales（市场和销售），26-28
oil slick detection（原油泄漏的探测），23
preventive maintenance of electromechanical devices（预防性维护的电子机械装置），25-26
scientific applications（科学方面的应用），28
file format converters（文件格式转换器），380-382
file mining（文件挖掘），49
filter（过滤器），290
filter in Weka（Weka中的过滤器），382-384
FilteredClassifier，401，414
filtering algorithms in Weka（Weka中的过滤算法），393-403
sparse instances（稀疏实例），401
supervised filters（有指导过滤器），401-403
unsupervised attribute filters（无指导属性过滤器），395-400
unsupervised instance filters（无指导实例过滤器），400-401
filtering approaches（过滤方法），315
filters menu（过滤器菜单），383
finite mixture（有限混合），262，263
FirstOrder，399
Fisher，R.A.，15
flat file（平文件），45
F-measure（F测量），172
FN（false negatives）（错误的否定），162
folds（折），150
forward pruning（正向修剪），34，192
forward selection（正向选择），292，294
forward stagewise additive modeling（正向阶段性叠加建模），325-327
Fourier analysis（傅里叶分析），25
FP（false positives）（错误的肯定），162
freedom（自由），degrees of（度）93，155
functional dependencies（函数相关性），350
functions in Weka（Weka具有的功能），404-405，409-410

G

gain ratio（增益率），104
GainRatioAttributeEval，423
gambling（赌博），160
garbage in，garbage out. *See* cost of errors; data cleaning; error rate
Gaussian-distribution assumption（高斯分布假设），92
Gaussian kernel function（高斯核函数），252
generalization as search（以搜索为目的概括），30-35
bias（偏差），32-35
enumerating concept space（枚举概念空间），31-32
generalized distance functions（经推广的距离函数），241-242
generalized exemplars（经推广的样本集），236
general-to-specific search bias（从一般到具体的搜索偏差），34
genetic algorithms（遗传算法），38
genetic algorithm search procedures（遗传算法搜索程序），294，341
GeneticSearch，424
getOptions ( )，482
getting to know your data（认识数据），60
global discretization（全局离散），297
globalInfo ( )，472
global optimization（全局优化），205-207
Gosset，William，184
gradient descent（梯度下降法），227，229，230
Grading，417
graphical models（图解模型），283
GraphViewer，431
gray bar in margin of textbook（optional sections）（正文旁边的灰条（选读部分）），30
greedy search（贪心搜索），33
GreedyStepwise，423-424
growing set（成长集），202

H

Hamming distance（汉明距离），335
hand-labeled data（手工标签数据），338
hapax legomena（罕用语），310
hard instances（难处理的实例），322

hash table ((散列表), 280
hazard detection system (危险探测系统), 23-24
hidden attributes (隐蔽属性), 272
hidden layer (隐蔽层), 226, 231, 232
hidden units (隐蔽单元), 226, 231, 234
hierarchical clustering (分级的聚类), 139
highly-branching attribute (高度分支的属性), 86
high-performance rule inducers (高性能规则归纳器), 188
histogram equalization (直方图均衡化), 298
historical literary mystery (历史文献之谜), 358
holdout method (旁置法), 146, 149-150, 333
homeland defense (国家防御), 357
HTML, 355
hypermetrope (远视), 13
hyperpipes (超管道), 139
Hyperpipes, 414
hyperplane (超平面), 124, 125
hyperrectangle (超矩形), 238-239
hyperspheres (超球面), 133
hypertext markup language (超文本结构语言) (HTML), 355
hypothesis testing (测试假说), 29

I

IB1, 413
IB3, 237
IBk, 413
ID3, 105
Id3, 404
identification code (标识码), 86, 102-104
implementation—real-world schemes (实现——真正的方案) 187-283
Bayesian networks (贝叶斯网络), 271-283
classification rules (分类规则), 200-214
clustering (聚类), 254-271
decision tree (决策树), 189-199
instance-based (基于实例的), 236-243
linear models (线性模型), 214-235
implementation—real-world schemes (continued)
numeric prediction (数值预测), 243-254
See also individual subject headings
inaccurate values (不正确的值), 59-60. *See also* cost of errors; data cleaning; error rate
incremental algorithms (递增算法), 346
incrementalClassifier, 434
IncrementalClassifierEvaluator, 431
incremental clustering (递增聚类), 255-260
incremental learning in Weka (在Weka中的递增学习), 433-435
incremental reduced-error pruning (递增减少误差修剪法), 203, 205
independent attributes (独立属性), 267
index (), 472
induction (归纳), 29
inductive logic programming (归纳逻辑编程), 48, 60, 75, 351
induct system (归纳系统), 214
industrial usage (工业用途) . *See* implementation—real-world schemes
inferring rudimentary rules (推断基本规则), 84-88
InfoGainAttributeEval, 422-423
informational loss function (信息损失函数), 159-160, 161
information-based heuristic (基于信息的探索), 201
information extraction (信息提炼), 354
information gain (信息增益), 99
information retrieval (信息检索), 171
information value (信息值), 102
infrequent words (不频繁的词), 353
inner cross-validation (内部交叉验证), 286
input (输入), 41-60
ARFF format (ARFF格式), 53-55
assembling the data (整合数据), 52-53
attribute (属性), 49-52
attribute types (属性类型), 56-57
concept (概念), 42-45
data engineering (数据处理), 286-287, 288-315. *See also* engineering input and output
data preparation (数据准备), 52-60
getting to know your data (了解数据), 60
inaccurate values (不正确的值) ,59-60
instances (实例), 45
missing values (残缺值), 58
sparse data (稀疏数据), 55-56
input layer (输入层), 224
instance in Weka (Weka中的实例), 450
Instance, 451
instance-based learning (基于实例的学习), 78, 128-136, 235-243
ball tree (球树), 133-135
distance functions (距离函数), 128-129, 239-242

finding nearest neighbors（寻找最近邻居），129-135
generalized distance functions（经推广的距离函数），241-242
generalized exemplars（经推广的样本集），236
k D-tree （*k* D树），130-132
missing values （残缺值），129
pruning noisy exemplars（修剪干扰样本），236-237
redundant exemplars（重复样本），236
simple method（简单方法），128-136，235-236
weighting attributes（加权属性），237-238
Weka，413-414
instance-based learning methods（基于实例的学习方法），291
instance-based methods（基于实例的方法），34
instance-based representation（基于实例的表达），76-80
instance filters in Weka（Weka中的实例过滤器），394，400-401，403
instances（实例），45
Instances，451
instance space（实例空间），79
instance weights（实例权值），166，321-322
integer-valued attributes（整数值属性），49
intensive care patients（危重病人），29
interval（区间），88
interval quantities （区间值），50-51
intrusion detection systems（入侵侦察系统），357
invertSelection，382
in vitro fertilization（人工受精），3
iris dataset（iris 数据集），15-16
iris setosa，15
iris versicolor，15
iris virginica，15
ISO-8601 combined date and time format（ISO-8601标准日期和时间的组合），55
item（项），113
item sets（项集），113，114-115
iterative distance-based clustering（基于距离的迭代聚类），137-138

J

J4.8，373-377
J48，404，450
Javadoc indices（Java文本索引），456
JDBC database，445
JRip，409
junk email filtering（垃圾邮件过滤），356-357

K

K2，278
Kappa statistic（Kappa 统计量），163-164
k D-trees（*k* D树），130-132，136
Kepler's three laws of planetary motion（开普勒的三条行星运行定律），180
kernel （核）
defined（定义的），235
perceptron（感知器），223
polynomial（多项式），218
RBF，219
sigmoid（S形），219
kernel density estimation（核密度估计），97
kernel logistic regression（核logistic 回归），223
kernel perceptron（核感知器），222-223
k-means（*k* 均值），137-138
k-nearest-neighbour method（*k* 最近邻方法），78
Knowledge Flow interface（知识流界面），427-435
configuring / connecting components（配置/连接组件），431-433
evaluation components（评估组件），430，431
incremental learning（递增学习），433-435
starting up（开始），427
visualization components（视觉化组件），430-431
knowledge representation（知识表达），61-82
association rules（关联规则），69-70. *See also* association rules
classification rules（分类规则），65-69. *See also* classification rules
clusters（聚类），81-82. *See also* clustering
decision table（决策表），62
decision tree（决策树），62-65. *See also* decision tree
instance-based representation（基于实例的表达），76-80
rules involving relations（包含关系的规则），73-75
rules with exceptions（带有例外的规则），70-73，210-213
trees for numeric prediction（用于数值预测的树），76
KStar，413

L

labor negotiations data（劳资协商数据），17-18，19
language bias（语言偏差），32-33
language identification （语言识别），353

Laplace, Pierre（拉普拉斯），91
Laplace estimator（拉普拉斯估计器），91，267，269
large datasets（大型数据集），346-349
law of diminishing returns（收益递减法则），347
lazy classifiers in Weka（Weka中的懒惰分类器），405，413-414
LBR，414
learning（学习），7-9
learning algorithms in Weka（Weka中的学习算法），403-404
  algorithm（算法），listed，404-405
  Bayesian classifier（贝叶斯分类器），403-406
  functions（函数），404-405，409-410
  lazy classifiers（懒惰分类器），405，413-414
  miscellaneous classifiers（杂项类分类器），405，414
  neural network（神经网络），411-413
  rules（规则），404，408-409
  trees（树），404，406-408
learning rate（学习率），229-230
least-absolute-error regression（最小绝对误差回归），220
LeastMedSq，409-410
leave-one-out cross-validation（留一交叉验证法），151-152
levels of measurement（测量标准），50
level-0 model（0层模型），332
level-1 model（1层模型），332
Leverage，420
Lift，420
lift chart（上升图），166-168，172
lift factor（上升系数），166
linear classification（线性分类），121-128
linearly separable（可线性分隔的），124
linear machine（线性机器），142
linear models（线性模型），119-128，214-235
  backpropagation（反向传播法），227-233
  computational complexity（计算复杂度），218
  kernel perceptron（核感知器），222-223
  linear classification（线性分类），121-128
  linear regression（线性回归），119-121
  logistic regression（logistic 回归），121-125
  maximum margin hyperplane（最大边际超平面），215-217
  multilayer perceptrons（多层感知器），223-226，231,233
  nonlinear class boundaries（非线性类边界），217-219
  numeric prediction（数值预测），119-120
  overfitting（过度拟合），217-218
  perceptron（感知器），124-126
  RBF network（RBF网络），234
  support vector regression（支持向量回归），219-222
  Winnow，126-128
linear regression（线性回归），77，119-121
LinearRegression，387，409
linear threshold unit（线性阈值单元），142
listOptions ()，482
literary mystery（文献之谜），358
LMT，408
load forecasting（负载预测），24-25
loan application（借贷应用），22-23
local discretization（局部离散），297
locally weighted linear regression（局部加权线性回归），244，251-253，253-254，323
locally weighted Naïve Bayes（局部加权朴素贝叶斯），252-253
Log button（Log按钮），380
logic programs（逻辑程序），75
logistic model trees（logistic 模型树），331
logistic regression（logistic 回归），121-125
LogitBoost，328，330，331
LogitBoost，416
logit transformation（对数变换），121
log-likelihood（对数-似然），122-123，276，277
log-normal distribution（对数-正态分布），268
log-odds distribution（对数-概率分布），268
LWL，414

M

M5' program，384
M5P，408
M5Rules，409
machine learning（机器学习），6
main ()，453
majority voting（多数投票），343
MakeDensityBasedClusterer，419
MakeIndicator，396，398
makeTree ()，472，480
Manhattan metric（曼哈顿度量），129
manufacturing processes（生产过程），28
margin（边差），324
margin curve（边差曲线），324
market basket analysis（购物篮分析），27

market basket data（购物篮数据），55
marketing and sales（市场和销售），26-28
Markov blanket （马尔可夫毯），278-279
Markov network （马尔可夫网络），283
massive datasets（大型数据集），346-349
maximization（最大化），265，267
maximum margin hyperplane（最大边际超平面），215-217
maxIndex ( )，472
MDL metric（MDL 衡量），277
MDL principle（MDL 原理），179-184
mean absolute error（平均绝对误差），177-179
mean-squared error（均方误差），177，178
measurement errors（测量误差），59
membership function（从属关系函数），121
memorization（记忆），76
MergeTwoValues，398
merging（合并），257
MetaCost，319，320
MetaCost，417
metadata （元数据），51，349，350
metadata extraction（元数据提炼），353
metalearner （元学习器），332
metalearning algorithms in Weka （Weka中的元学习算法），414-418
metric tree（测量树），136
minimum description length （MDL） principle（最短描述长度原理），179-184
miscellaneous classifiers in Weka（Weka 中的杂项类分类器），405，414
missing values（残缺值），58
classification rules（分类规则），201-202
decision tree（决策树），63，191-192
instance-based learning（基于实例的学习），129
1R，86
mixture model（混合模型），267-268
model tree（模型树），246-247
statistical modeling（统计建模），92-94
mixed-attribute problem（混合属性问题），11
mixture model（混合模型），262-264，266-268
MLnet，38
ModelPerformanceChart，431
model tree（模型树），76，77，243-251
building the tree（建树），245
missing values（残缺值），246-247
nominal attributes（名词性属性），246
pruning（修剪），245-246
pseudocode（伪代码），247-250
regression tree induction（回归树归纳），compared，243
replicated subtree problem（复制子树问题），250
rules（规则），250-251
smoothing（平滑），244，251
splitting（分裂），245，247
what is it（它是什么），250
momentum（动量），233
monitoring，continuous（监测，连续的），28-29
MultiBoostAB，416
multiclass alternating decision trees，(多类交互式决策树) 329，330，343
MultiClassClassifier，418
multiclass learning problems（多类学习问题），334
MultilayerPerceptron，411-413
multilayer perceptrons（多层感知器），223-226，231，233
multinomial distribution（多项分布），95
multinomial Naïve Bayes（多项朴素贝叶斯），95，96
multiple linear regression（多重线性回归），326
multiresponse linear regression（多反馈线性回归），121，124
multistage decision property（多级的决策属性），102
multivariate decision trees（多元决策树），199
MultiScheme，417
myope（近视），13

N

NaiveBayes（朴素贝叶斯），403，405
Naïve Bayes（朴素贝叶斯），91，278
clustering for classification（用于分类的聚类），337-338
co-training（联合训练），340
document classification（文件分类），94-96
limitations（局限），96-97
locally weighted （局部加权的），252-253
multinomial（多项的），95，96
power（能力），96
scheme-specific attribute selection（特定方案的属性选择），295-296
selective（选择性的），296
TAN （Tree Augmented Naïve Bayes） （树扩展型朴素贝叶斯），279
what can go wrong（产生错误），91

NaiveBayesMultinominal, 405
NaiveBayesSimple, 403
NaiveBayesUpdateable, 405
NBTree, 408
nearest-neighbor learning（最近邻学习）, 78-79, 128-136, 235, 242
nested exceptions（嵌套的例外）, 213
nested generalized exemplars（嵌套推广样本集）, 239
network scoring（网络评分）, 277
network security（网络安全）, 357
neural networks（神经网络）, 39, 233, 235, 253
neural networks in Weka（Weka 中的神经网络）, 411-413
n-gram profiles（*n*-gram 文件轮廓）, 353, 361
Nnge, 409
noise（干扰）
  data cleansing（数据清理）, 312
  exemplars（样本集）, 236-237
  hand-labeled data（手工标签数据）, 338
  robustness of learning algorithm（学习算法的稳健度）, 306
noisy exemplars（干扰样本）, 236-237
nominal attributes（名词性属性）, 49, 50, 56-57, 119
  Cowweb, 271
  convert to numeric attributes（转换成数值属性）, 304-305
  decision tree（决策树）, 62
  mixture model（混合模型）, 267
  model tree（模型树）, 246
  subset（子集）, 88
nominal quantities（名词性值）, 50
NominalToBinary, 398-399, 403
non-axis-parallel class boundaries（非轴平行类边界）, 242
Non-Bayesians（非贝叶斯）, 141
nonlinear class boundaries（非线性类边界）, 217-219
NonSparseToSparse, 401
normal-distribution assumption（正态分布假设）, 92
normalization（正常化）, 56
Normalize, 398, 400
normalize (), 480
normalized expected cost（经正常化的期望成本）, 175
nuclear family（核心家庭）, 47
null hypothesis（零假设）, 155
numeric attribute（数值属性）, 49, 50, 56-57
  axis-parallel class boundaries（轴平行类边界）, 242
  classification rules（分类规则）, 202
  Classit, 271
  converting discrete attributes to（将离散属性转换为）, 304-305
  decision tree（决策树）, 62, 189-191
  discretizing（离散）, 296-305. *See also* Discretizing numeric attributes
  instance-based learning（基于实例的学习）, 128, 129
  interval（区间）, 88
  linear models（线性模型）, 119
  linear ordering（线性排序）, 349
  mixture model（混合模型）, 268
  1R, 86
  statistical modeling（统计建模）, 92
numeric-attribute problem（数值属性问题）, 11
numeric prediction（数值预测）, 43-45, 243-254
  evaluation（评估）, 176-179
  forward stagewise additive modeling（正向阶段迭加建模）, 325
  linear regression（线性回归）, 119-120
  locally weighted linear regression（局部加权线性回归）, 251-253
numeric prediction（数值预测）(continued)
  model tree（模型树）, 244-251. *See also* model tree
  rules（规则）, 251
  stacking（堆栈）, 334
  trees（树）, 76, 243
NumericToBinary, 399
NumericTransform, 397

O

O (n), 196
O (n^2), 196
Obfuscate, 396, 400
object editor（对象编辑器）, 366, 381, 393
Occam's razor（Occam剃刀）, 180, 183
oil slick detection（浮油探测）, 23
1R procedure（1R 程序）, 84-88, 139
OneR, 408
OneRAttributeEval, 423
one-tailed（单边）, 148
online documentation（在线文件）, 368
Open DB, 382
optimizing performance in Weka（在Weka 中优化性能）, 417
OptionHandler, 451, 482

option nodes（选择节点），328
option trees（选择树），328-331
orderings （排序）
circular（循环的），349
partial（局部的），349
order-independent rules（顺序独立规则），67，112
OrdinalClassClassifier，418
ordinal attributes（有序属性），51
ordinal quantities（有序值），50
orthogonal（正交的），307
outer cross-validation（外层交叉验证），286
outliers（孤立点），313，342
output（输出）
data engineering（数据处理），287-288，315-341. *See also* engineering input and output
knowledge representation（知识表达），61-82. *See also* knowledge representation
overfitting（过度拟合），86
Bayesian clustering（贝叶斯聚类），268
category utility（类别效用），261
forward stagewise additive regression （正向阶段迭加回归），326
MDL principle（MDL原理），181
multilayer perceptrons（多层感知器），233
1R，87
statistical tests（统计测试），30
support vectors（支持向量），217-218
overfitting-avoidance bias（避免过度拟合偏差），34
overgeneralization（过度推广），239，243
overlapping hyperrectangles（重叠超矩形），239
overlay data（重叠数据），53

P

Pace regression in Weka（Weka中的Pace回归），410
PaceRegression，410
paired *t*-test（成对的 *t* 测试），154，294
pairwise classification（成对分类），123,410
pairwise coupling（成对合并），123
pairwise plots（成对属性值坐标投影），60
parabola（抛物线），240
parallelization（并行化），437
parameter tuning（参数调整），286
Part，409
partial decision tree（局部决策树），207-210
partial ordering（局部排序），51
partitioning instance space（分割实例空间），79
pattern recognition（模式识别），39
Percentage split，377
perceptron （感知器）
defined（定义的），126
kernel（核），223
learning rule（线性规则），124，125
linear classification（线性分类），124-126
multilayer（多层），223-226，233
voted（投票的），223
perceptron learning rule（感知器学习规则），124，125
permutation tests（排列测试），362
PKIDiscretize，396，398
Poisson distribution（泊松分布），268
Polygon，389
Polyline，389
polynomial kernel（多项式核），218
popular music（流行音乐），359
postal ZIP code （邮政编码），57
postpruning （后修剪），34,192
precision （精确率），171
predicate calculus（谓词验算），82
predicting performance（预测性能），146-149. *See also* evaluation
predicting probabilities（预测概率），157-161
PredictionAppender，431
prediction nodes（预测节点），329
predictive accuracy in Weka（Weka中的预测正确率），420
PredictiveApriori，420
Preprocess panel，372，380
prepruning （前修剪），34,192
presbyopia（远视眼），13
preventive maintenance of electromechanical devices（预防性维护的电子机械装置），25-26
principal components（主分量），307-308
PrincipalComponents，423
principal components analysis（主分量分析），306-309
principle of multiple explanations （多种解释原理），183
prior knowledge（先验知识），349-351
prior probability（先验概率），90
PRISM，110-111，112，213
Prism，409
privacy （隐私），357-358
probabilistic EM procedure（概率EM程序），265-266
probability-based clustering（基于概率的聚类），262-265

probability cost function（概率成本函数)，175
probability density function（概率密度函数)，93
programming （编程）. *See* Weka workbench
programming by demonstration （示范编程)，360
promotional offers（促销商品)，27
proportional k-interval discretization（均衡k区间离散)，298
propositional calculus （命题演算)，73，82
propositional rules（命题规则)，69
pruning（修剪）
classification rules（分类规则)，203，205
decision tree（决策树)，192-193，312
massive datasets（大型数据集)，348
model tree（模型树)，245-246
noisy exemplars（干扰实例)，236-237
overfitting-avoidance bias（避免过度拟合偏差)，34
reduced-error（减少-误差)，203
pruning set（修剪集)，202
pseudocode（伪代码）
basic rule learner（基本规则学习器)，111
model tree（模型树)，247-250
1R，85
punctuation conventions（标点符号习惯)，310

Q

quadratic losss function（二次损失函数)，158-159，161
quadratic optimization（二次优化)，217
Quinlan，J.Ross，29，105，198

R

R.R.Donnelly，28
RacedIncrementalLogitBoost，416
race search（竞赛搜索)，295
RaceSearch，424
radial basis function （RBF） kernel（径向基核函数)，219，234
radial basis function（RBF）network（径向基函数网络)，234
RandomCommittee，415
RandomForest，407
random forest metalearner in Weka，416
randomization（随机化)，320-321
Randomize，400
RandomProjection，400
random projections（随机投影)，309
RandomSearch，424
RandomTree，407
Ranker，424-425
RankSearch，424
ratio quantities（比率值)，51
RBF （Radial Basis Function） kernel（径向基核函数)，219,314
RBF （Radial Basis Function） network （径向基函数网络)，234
RBFNetwork，410
real-life applications（实际应用）. *See* fielded applications
real-life datasets（现实生活中的数据集)，10
real-world implementations（真正实践中的实现）. *See* implementation—— real-world schemes
recall（反馈率)，171
recall-precision curves（反馈率-精确率曲线)，171-172
Rectangle，389
rectangular generalizations（矩形推广)，80
recurrent neural networks（循环神经网络)，233
recursion（递归)，48
recursive feature elimination （递归属性消除)，291，341
reduced-error pruning （减少-误差修剪法)，194，203
redundant exemplars（重复实例)，236
regression（回归)，17，76
RegressionByDiscretization，418
regression equation（回归等式)，17
regression tree（回归树)，76，77，243
reinforcement learning（增强学习)，38
relational data（关系数据)，49
relational rules（关系规则)，74
relations（关系)，73-75
relative absolute error（相对绝对误差)，177-179
relative error figures（相对误差值)，177-179
relative squared error（相对平方误差)，177，178
RELIEF，341
ReliefFAttributeEval，422
religious discrimination，illegal（性别歧视，非法)，35
remoteEngine.jar，446
remote.policy，446
Remove，382
RemoveFolds，400
RemovePercentage，401
RemoveRange，401
RemoveType，397

RemoveUseless, 397
RemoveWithValues, 401
repeated holdout（重复旁置）, 150
ReplaceMissingValues, 396, 398
replicated subtree problem（重复子树问题）, 66-68
REPTree, 407-408
Resample, 400, 403
residuals（残差）, 325
resubstitution error（重新代入误差）, 145
Ridor, 409
RIPPER rule learner, 205-214
ripple-down rules（ripple-down规则）, 214
robo-soccer（机器人足球）, 358
robust regression（稳健回归）, 313-314
ROC curve（ROC曲线）, 168-171, 172
root mean-squared error（均方根误差）, 178, 179
root relative squared error（相对平方根误差）, 178, 179
root squared error measures（平方根误差衡量）, 177-179
rote learning（机械的学习方法）, 76, 354
row separation（行分隔）, 336
rule（规则）
  antecedent（前提）, 65
  association（关联）, 69-70, 112-119
  classification（分类）. *See* classification rules
  consequent（结论）, 65
  decision lists（决策列）, 111-112
  double-consequent（两个结论）, 118
  exceptions（例外）, with, 70-72, 210-213
  good（worthwhile）（好的（值得的））, 202-205
  nearest-neighbor（最近邻）, 78-79
  numeric prediction（数值预测）, 251
  order of（decision list）（（决策列）的顺序）, 67
  partial decision trees（局部决策树）, 207-210
  propositional（命题的）, 73
  relational（关系的）, 74
  relations（关系）, and, 73-75
  single-consequent（一个结论）, 118
  trees（树）, and, 107, 198
  Weka, 408-409
rule-based programming（基于规则的程序）, 82
rules involving relations（包括关系的规则）, 73-75
rules with exceptions（带有例外的规则）, 70-73, 210-213

S

sample problems（取样问题）. *See* example problems
sampling with replacement（放回抽样）, 152
satellite images, evaluating（卫星照片，评估）, 23
ScatterPlotMatrix, 430
schemata search（模式搜索）, 295
scheme-independent attribute selection（独立于方案的属性选择）, 290-292
scheme-specific attribute selection（特定方案的属性选择）, 294-296
scientific applications（科学方面的应用）, 28
scoring networks（评分网络）, 277-280, 283
SDR（Standard Deviation Reduction）（标准差减少值）, 245
search bias（搜索偏差）, 33-34
search engine spam（搜索引擎垃圾）, 357
search methods in Weka（Weka中的搜索方法）, 421, 423-425
segment-challenge.arff, 389
segment-test.arff, 389
Select attributes panel, 392-393
selective Naïve Bayes（选择性的朴素贝叶斯）, 296
semantic relation（语义关系）, 349
semantic Web（语义网络）, 355
semisupervised learning（半指导学习）, 337
sensitivity（敏感性）, 173
separate-and-conquer technique（割治技术）, 112, 200
sequential boosting-like scheme（有序的类似提升的方法）, 347
sequential minimal optimization（SMO）algorithm（连续最小优化算法）, 410
setOptions(), 482
sexual discrimination, illegal（性别歧视，非法）, 35
shapes problem（几何图形问题）, 73
sigmoid function（S形函数）, 227, 228
sigmoid kernel（S形核函数）, 219
Simple CLI, 371, 449, 450
SimpleKMeans, 418-419
simple linear regression（简单线性回归）, 326
SimpleLinearRegression, 409
SimpleLogistic, 410
simplest-first ordering（简单优先顺序）, 34
simplicity-first methodology（简洁为先的方法）, 83, 183
single-attribute evaluators in Weka（Weka 中的单一属性评估器）, 421, 422-423
single-consequent rules（一个结论的规则）, 118
single holdout procedure（单次旁置过程）, 150

sister-of-relation（姊妹关系）、46-47
SMO、410
smoothing （平滑）
locally weighted linear regression （局部加权线性回归）、252
model tree（模型树）、244、251
SMOreg、410
software programs（软件编程）. *See* Weka workbench
sorting（排序）、avoiding repeated （避免重复）、190
soybean data（大豆数据）、18-22
spam（垃圾邮件）、356-357
sparse data（稀疏数据）、55-56
sparse instance in Weka（Weka中的稀疏数据）、401
SparseToNonSparse、401
specificity（特征）、173
specific-to-general search bias（具体到一般的搜索偏差）、34
splitData ()、480
splitter nodes （分裂节点）、329
splitting（分裂）
clustering（聚类）、254-255、257
decision tree（决策树）、62-63
entropy-based discretization（基于熵的离散）、301
massive datasets（大型数据集）、347
model tree（模型树）、245、247
subexperiments（子实验）、447
surrogate （代理）、247
SpreadSubsample、403
squared-error loss function（平方误差损失函数）、227
squared error measures（平方误差衡量）、177-179
stacked generalization（堆栈式推广）、332
stacking （堆栈）、332-334
Stacking、417
StackingC、417
stale data（失效的数据）、60
standard deviation reduction（SDR）（标准差减少值）、245
standard deviations from the mean（距离平均值标准差）,148
Standardize、398
standardizing（标准化）、56
statistical modeling（统计建模）、88-97
document classification（文件分类）、94-96
missing values（残缺值）、92-94
normal-distribution assumption（正态分布假设）、92
numeric attributes（数值属性）、92-94
statistics（统计学）、29-30
Status box、380
step functuion、227、228
stochastic algorithms（随机算法）、348
stochastic backpropagation（随机反向传播）、232
stopping criterion（停止标准）、293、300、326
stopwords（停止词）、310、352
stratification（分层）、149、151
stratified holdout（分层旁置）、149
StratifiedRemoveFolds、403
stratified cross-validation（分层交叉验证）、149
StreamableFilter、456
string attributes（字符串属性）、54-55
string conversion in Weka（Weka中的字符串转换）、399
string table（字符表）、55
StringToNominal、399
StringToWordVector、396、399、401、462
StripChart、431
structural patterns（结构模式）、6
structure learning by conditional independence test（采用条件独立测试的结构学习）、280
student's distribution with k－1 degrees of freedom（自由度为k-1的学生氏分布）、155
student's *t*-test （学生氏*t*测试）、154、184
subexperiments（子实验）、447
subsampling in Weka（Weka 中的子采样）、400
subset evaluators in Weka（Weka中的子集评估器）、421、422
subtree raising（子树上升）、193、197
subtree replacement（子树置换）、192-193、197
success rate （成功率）、173
supervised attribute filters in Weka（Weka中有指导属性过滤器）、402-403
supervised discretization（有指导离散）、297、298
supervised filters in Weka（Weka中的有指导过滤器）、401-403
supervised instance filters in Weka（Weka中有指导实例过滤器）、402、403
supervised learning（有指导学习）、43
support（支持）、69、113
support vector（支持向量）、216
support vector machine（支持向量机）、39、188、214、340
support vector machine （SVM） classifier（支持向量机分类器）、341
support vector machines with Gaussian kernels（使用高

斯核函数的支持向量机)，234
support vector regression（支持向量回归），219-222
surrogate splitting （代理分裂），247
SVMAttributeEval，423
SVM classifier （Support Vector Machine）（SVM 分类器（支持向量机）），341
SwapValues，398
SymmetricalUncertAttributeEval，423
symmetric uncertainty（对称不定性），291
systematic data errors（系统数据错误），59-60

T

tabular input format，119
TAN （Tree Augmented Naïve Bayes）（树扩展型朴素贝叶斯），279
television preferences / channels（受欢迎的电视频道），28-29
tenfold cross-validation（10折交叉验证），150，151
Tertius，420
test set（测试集），145
TestSetMaker，431
text mining（文本挖掘），351-356
text summarization（文本归纳），352
text to attribute vectors（从文本到属性向量），309-311
TextViewer，430
TF × IDF，311
theory（理论），180
threat detection systems（威胁侦察系统），357
3-point average recall （3点平均反馈率），172
threefold cross-validation（3折交叉验证），150
ThresholdSelector，418
time series（时间序列），311
TimeSeriesDelta，400
TimeSeriesTranslate，396，399-400
timestamp（时间戳），311
TN （True Negatives）（正确的否定），162
tokenization，310
tokenization in Weka，399
top-down induction of decision trees （自上而下的决策树归纳），105
toSource ( )，453
toString ( )，453，481，483
toy problems（玩具问题）. *See* example problems
TP （True Positives）（正确的肯定），162
training and testing（训练和测试），144-146
training set（训练集），296
TrainingSetMaker，431
TrainTestSplitMaker，431
transformations（转换）. *See* attribute transformations
transforming a multiclass problem into a two-class one （将多类问题转换为2类问题），334-335
tree（树）
  AD （All Dimensions），280-283
  alternating decision（交互式决策），329，330，343
  ball（球），133-135
  decision（决策）. *See* decision tree
  logistic model（logistic 模型），331
  metric（度量），136
  model（模型），76，243. *See also* model tree
  numeric prediction（数值预测），76
  option（选择），328-331
  regression（回归），76，243
Tree Augmented Naïve Bayes （TAN） （树扩展型朴素贝叶斯），279
tree classifier in Weka（Weka中的树分类器），404，406-408
tree diagrams（树图），82
Trees（subpackages），451，453
Tree Visualizer，389，390
true negative （TN）（正确的否定），162
true positive （TP）（正确的肯定），162
true positive rate（正确肯定率），162-163
True positive rate，378
t-statistic，156
t-test（*t*测试），154
TV preferences/channels（受欢迎的电视频道），28-29
two-class mixture model（二类混合模型），264
two-class problem（二类问题），73
two-tailed test（双边测试），156
two-way split（二叉分裂），63
typographic errors（印刷错误），59

U

ubiquitous data mining （无处不在的数据挖掘），358-361
unacceptable contracts（不能接受的合同），17
Unclassified instances，377
Undo，383
unit（单元），224
univariate decision tree（单变量决策树），199
universal language（通用语言），32
unlabeled data（无标签数据），337-341

clustering for classification（用于分类的聚类），337
co-training（联合训练），339-340
EM and co-training（EM和联合训练），340-341
unmasking（揭密），358
unsupervised attribute filters in Weka（Weka中无指导属性过滤器），395-400
unsupervised discretization（无指导离散），297-298
unsupervised instance filters in Weka（Weka中无指导实例过滤器），400-401
unsupervised learning（无指导学习），84
UpdateableClassifier，456，482
UpdateClassifier（），482
User classifier（用户自己的分类器），63-65，388-391
UserClassifier，388
user interfaces（用户界面），367-368
Use training set，377
utility（效用），category（类别），260-262

V

validation data（验证数据），146
variance（方差），154，317
Venn diagram（Venn图），81
very large datasets（大型数据集），346-349
"Very simple classification rules perform well on most commonly used datasets"（Holte）（简单的分类规则在多数通用的数据集上表现突出），88
VFI，414
visualization components in Weka（weka中的可视化组件），430-431
Visualize classifier errors，387
Visualize panel，393
Visualize threshold curve，378
Vote，417
voted perceptron（投票感知器），223
VotedPerceptron，410
voting（投票），315，321，347
voting feature intervals（投票特征区间），136

W

weak learners（弱学习器），325
weather problem example（天气问题实例），10-12
association rules for（……的关联规则），115-117
attribute space for（……的属性空间），292-293
as a classification problem（作为一个分类问题），42
as a clustering problem（作为一个聚类问题），43-44
converting data to ARFF format（将数据转换为ARFF文件格式），370
cost matrix for（……的成本矩阵），457
evaluating attributes in（评估……的属性），85-86
infinite rules for（有限的规则），30
item sets（项集），113-115
as a numeric prediction problem（作为一个数值预测问题），43-44
web mining（网络挖掘），355-356
weight decay（权衰减），233
weighted instances（加权实例），252
WeightedInstancesHandler，482
weighting attributes（属性加权），237-238
weighting models（模型加权），316
weka.associations，455
weka.attributeSelection，455
weka.classifiers，453
weka.classifiers.bayes.naiveBayesSimple，472
weka.classifiers.Classifier，453
weka.classifiers.lazy.IB1，472
weka.classfiers.lazy.IBk，482，483
weka.classifiers.rules.Prism，472
weka.classifiers.trees，453
weka.classifiers.trees.Id3，471，472
weka.clusterers，455
weka.core，451，452，482-483
weka.estimators，455
weka.filters，455
weka workbench（Weka工作平台），365-483
class hierarchy（类的层级），471-483
classifiers（分类器），366，471-483
command-line interface（命令行界面），449-459.
See also command-line interface
elementary learning schemes（基本学习方案），472
embedded machine learning（嵌入式机器学习），461-469
example application（classify text files into two categories）示范程序（将文本文件划分为两个类别），461-469
Experimenter（实验者），437-447
Explorer（探索者），369-425. *See also* Explorer
implementing classifiers（分类器），471-483
introduction（介绍），365-368
Knowledge Flow interface（知识流界面），427-435
neural-network GUI（神经网络图形用户界面），411
object editor（对象编辑器），366

online documentation（在线文本），368
user interfaces（用户界面），367-368
William of Occam，180
Winnow，410
Winnow algorithm（Winnow 算法），126-128
wisdom，defined（智慧，定义），37
Wolpert，David，334
word conversions（单词转换），310
World Wide Web mining（网络挖掘），354-356
wrapper（包装），290，341，355
wrapper induction （包装归纳），355
WrapperSubsetEval，422
writing classifiers in Weka（在Weka中编写分类器），471-483

Z

0-1 loss function（0-1损失函数），158
0.632 bootstrap（自引导法），152
1R method（单规则方法），84-88
zero-frequency problem（零频率问题），160
zero point，inherently defined（零点，内在定义），51
ZeroR，409
ZIP code（邮编），57